Neuvotteluhuoneiden varaustilannetta tarkkailevan pluginin kehittäminen Confluence-järjestelmään

Case: Ambientia Group Oy

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Hämeenlinna, Kevät 2018

Joni Koivula

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely
Hämeenlinna

Tekijä Joni Koivula **Vuosi** 2018

Työn nimi Neuvotteluhuoneiden varaustilannetta tarkkailevan pluginin kehittäminen Confluence-järjestelmään

Työn ohjaaja/t Tommi Lahti

TIIVISTELMÄ

Opinnäytetyön aiheena on suunnitella ja toteuttaa asiakkaalle lisäosa eli plugin Atlassianin Confluence-järjestelmään. Plugin tarkkailee asiakkaan toimiston neuvotteluhuoneiden varaustilanteita käyttämällä Zimbra-järjestelmän REST-ohjelmointirajapintaa neuvotteluhuoneiden kalenterien tietojen noutamiseen. Opinnäytetyön aiheen toimeksianto tuli asiakkaalta, Ambientia Group Oy:ltä.

Työ alkaa aiheeseen liittyvien teknologioiden ja järjestelmien esittelyllä, josta siirrytään Atlassian SDK -kehittäjäpaketin työkalujen käytön demonstrointiin esimerkki plugin projektilla. Työn teoriataustan ja Atlassian SDK -kehittäjäpaketin työkalujen käytön esimerkeistä siirrytään asiakkaan pluginin suunnitteluun ja toteutukseen.

Pluginin kehitystyön kerrontaa käydään läpi käytännön läheisesti ja toteutuksen jokainen osa-alue keskittyy pluginin suunnittelussa määritettyjen keskeisen ominaisuuden tai toiminnallisuuden toteuttamiseen. Tämän työn keskeisenä tavoitteena on selventää, mikä on Confluence-järjestelmän plugin ja miten Atlassian SDK -kehittäjäpaketin työkaluja käytetään pluginin kehitystyössä.

Asiakkaalle kehitetty Confluence plugin on tämän työn aikana toteutettu suunnittelussa määritetyiltä toiminnallisuuksiltaan täysin. Pluginin suunnittelu ja toteutus onnistuivat hyvin ja työlle määrätyn aikataulun puitteissa. Yhteistyössä asiakkaan vastuuhenkilön kanssa plugin on toteutettu tämän työn pohjalta, lukuun ottamatta käyttöönottoa, asennusta ja testausta asiakkaan Confluence-järjestelmässä.

Avainsanat Confluence, Plugin, Ohjelmointi, Java, JavaScript

Sivut 45 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna

Author Joni Koivula **Year** 2018

Subject Development of a Confluence plugin that tracks meeting rooms reservation status.

Supervisors Tommi Lahti

ABSTRACT

The subject of this thesis is to design and develop a plugin for the customer's Atlassian Confluence platform. The plugin tracks the customer's meeting room reservation status by using Zimbra system's REST API methods which retrieve Zimbra's calendar information. The thesis subject was commissioned by Ambientia Group Oy.

The thesis begins by getting familiar with the subject related technologies and systems, such as Atlassian Confluence, Zimbra and REST APIs in general. Afterwards, a closer examination is given to Atlassian SDK tools by the customer's request. The examination is a quick tutorial to the Atlassian SDK basic tools and a small plugin project. From the tutorial, the thesis moves to the designing and development of the customer commissioned Confluence plugin.

The plugin development chapter takes a very practical approach to the development of each functionality, which was defined as the plugin requirements in the plugin design chapter. The core goals of this thesis are to inform what a Confluence plugin is and how to use the Atlassian SDK tools in the development of Confluence plugins.

Customer commissioned Confluence plugin development is completed in this thesis work apart from the installment and testing in the customer's Confluence platform.

Keywords Confluence, Plugin, Programming, Java, JavaScript

Pages 45 pages

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on suunnitella ja toteuttaa asiakkaalle lisäosa eli plugin, Atlassianin Confluence-järjestelmään. Plugin tarkkailee asiakkaan toimiston neuvotteluhuoneiden varaustilanteita käyttämällä Zimbra-järjestelmän REST-ohjelmointirajapintaa neuvotteluhuoneiden kalenterien tietojen noutamiseen. Opinnäytetyön aiheen toimeksianto tuli asiakkaalta, Ambientia Group Oy:ltä. Ambientia on hämeenlinnalainen yritys, joka tarjoaa palveluja ja ratkaisuja yritysten digitalisointiin, palvelumuotoiluun ja softakehitykseen. Ambientia on tehnyt yhteistyötä Atlassianin kanssa jo kymmenen vuotta ja kuuluu Atlassianin korkeimpaan kumppanuustasoon. Ambientian Atlassialle tuottamat järjestelmien ratkaisut ja palvelut painottuvat IT-palveluhallinnointiin.

Opinnäytetyö aloitetaan lähestymisellä aiheeseen liittyvän teknologioiden ja tekniikoiden teorialla. Tutustun Atlassianiin ja Confluence-järjestelmään ja tarkemmin siihen, mikä on Confluence-järjestelmän plugin ja mistä se rakentuu. Käyn läpi Zimbra-järjestelmän REST-ohjelmointirajapintaa ja REST-ohjelmointirajapinnan periaatteita ja käyttöä ohjelmoinnissa. Ennen pluginin toteutuksen aloittamista, tutustutaan Atlassian SDK -kehittäjäpaketin käyttämiseen Atlassianin järjestelmien plugin-kehittämiseen. Viimeisenä siirrytään asiakkaan pluginin suunnittelun ja toteutuksen prosessien läpikäymiseen.

Työn keskeisinä tavoitteina on vastata kysymyksiin:
Mikä on Confluence-järjestelmän lisäosa eli plugin?
Miten Atlassian SDK -kehittäjäpaketin työkaluja käytetään plugin kehitystyössä?

2 ATLASSIAN CONFLUENCE

Atlassian on vuonna 2002 perustettu australialainen yritysohjelmistoja tuottava yritys, joka kehittää ohjelmistoja ja työkaluja ohjelmistokehityksen ja tiimityöskentelyn tueksi. Atlassianin kattava tuoteperhe tarjoaa ohjelmistoja ja palveluita muun muassa projektinhallintaan, organisaation sisäiseen viestintään, ohjelmistojen versionhallintaan ja tehtävien seurantaan. (Bloomberg, n.d.)

2.1 Confluence

Confluence on Atlassianin kehittämä tiimityöskentelyohjelmisto. Confluence-järjestelmää käytetään perusviestintään, projektinhallintaan ja sosiaaliseen verkostoitumiseen. Confluence-järjestelmän tavoitteena on ollut luoda työtiimeille keskitetty työympäristö, jonka avulla työtiimissä työskentely on vaivattomampaa ja nopeampaa. Confluencen avulla työtiimit voivat yhdessä luoda projekteja, jakaa dokumentteja ja kommunikoida yhdessä keskitetyssä ympäristössä. Confluence on keskeinen osa Atlassianin kehittämää laajaa tuoteperhettä ja on integroitavissa yrityksen muiden ohjelmistojen kanssa. (Atlassian n.d.a.)

Ideana on, että yrityksen jokaiselle merkittävälle osastolle ja projektille luodaan oma sivu, jonka avulla osastoon tai projektiin kuuluva henkilöstö kykenee vaivattomasti työskentelemään yhdessä keskitetyssä ympäristössä. Sivuille henkilöstö voi luoda ja tuoda lähes mitä tahansa tietoa, kuten muistiinpanoja, projektisuunnitelmia ja mediaa. Käyttäjät voivat antaa välittömästi kommentteja kaikkeen toisten käyttäjien luomaan sisältöön, mikä tekee työskentelytiimin kommunikoinnista tehokkaampaa (Atlassian n.d.b.). Confluence-järjestelmään on tarjolla yli 800 lisäsovellusta, joiden avulla Confluence-järjestelmä voidaan kustomoida ja räätälöidä palvelemaan yrityksen tarpeita. (Atlassian n.d.b.)

Confluence-järjestelmä on web-sovellus, joten käyttäjät tarvitsevat ainoastaan yhteensopivan, modernin selaimen. Tämä tarkoittaa, että käyttäjät voivat käyttää Confluence-järjestelmää usealla eri päätelaitteella, tietokoneella tai mobiililaitteella. Confluence on erittäin riippuvainen moderneista web-teknologioista ja -standardeista, joten vanhemmat selaimet tai versiot selaimista eivät välttämättä ole yhteensopivia Confluence-järjestelmän kanssa. (Kohler 2013, 2.)

2.2 Atlassian SDK -kehittäjäpaketti

SDK eli Software Development Kit on kokoelma ohjelmistoja, joita käytetään ohjelmistokehityksen työkaluina tietyille laitteille, alustoille tai käyttöjärjestelmille. Ohjelmistoympäristön asennuksen yhteydessä asennetaan tyypillisesti useita SDK-kehittäjäpaketteja. SDK-kehittäjäpaketit ovat

usein ilmaisia, koska yritykset haluavat kannustaa käyttäjiä kehittämään heidän järjestelmiään. (Sharpened Productions n.d.)

Jotta käyttäjät voivat aloittaa lisäosan eli pluginin kehittämisen Confluence-järjestelmään, tai muihin Atlassianin järjestelmiin, heidän täytyy asentaa työympäristöönsä Atlassian SDK -kehittäjäpaketti (Atlassian n.d.e.). Kehittäjäpaketti tarjoaa kokoelman komentokehotetyökaluja, joilla kehittäjät voivat luoda, asentaa ja rakentaa uusia Atlassianin järjestelmien plugin projekteja. Atlassian SDK -kehittäjäpaketin työkaluilla luodaan uusi plugin projektipohja, joka on määritetty luonnin yhteydessä yhteensopivaksi halutulle Atlassian järjestelmälle. Työkaluilla voidaan ajaa Atlassianin järjestelmien Demonstration Space -demonstrointitiloja, joissa plugineja voidaan testata. (Atlassian, n.d.k.)

3 REST-OHJEMOINTIRAJAPINTA

REST on Roy Fieldingn vuonna 2000 kuvaama ohjelmistokehityksen arkkitehtuurityyli. REST eli Representational State Transfer, ei ole itsessään ohjelmistokehityksen arkkitehtuurimalli, vaan kokoelma rajoituksia ja ohjeita ohjelmiston suunnitteluun. REST-mallin ohjeet ja rajoitukset kuvaavat tiettyjä rooleja datalle, komponenteille, hyperlinkeille, kommunikointiprotokollille ja datan kuluttajille. (Sandoval 2009, 24.)

Fielding kehitti REST- arkkitehtuurityylin tutkittuaan saatavilla olevia verkkoresursseja ja teknologioita hajautettujen järjestelmien kehittämiseen. Ilman rajoituksia ja ohjeita kehitetään vain enemmän vaikeasti hallinoitavia ja laajennettavia sovelluksia. REST-tyylisen (RESTful) sovelluksen tulee olla asiakas-palvelin-mallinen, tilaton (eli sovellus ei seuraa käyttäjän sessioita eli jokaisen palvelimelle tulevan pyynnön tulee olla itsenäinen), yhtenäinen resurssien saatavuuden kannalta ja skaalautuva. (Sandoval 2009, 24.)

Fielding REST- arkkitehtuurityylissä kuvatut rajoitteet ja ohjeet eivät määrää käytettävää teknologiaa. Rajoitteet ja ohjeet määräävät vain, kuinka ja millä tavoin dataa käsitellään komponenttien välillä ja mitkä ovat näiden rajoitteiden hyödyt. Tämä tarkoittaa, että REST- arkkitehtuurityyli voidaan implementoida mihin tahansa ohjelmistoarkkitehtuurimalliin. (Sandoval 2009, 25.)

REST-arkkitehtuurityylisiä web-sovelluksia kutsutaan RESTful-tyylisiksi sovelluksiksi.

3.1 Resurssit ja niiden esittäminen

RESTful-tyylinen resurssi on mitä tahansa, mihin voidaan osoittaa web-osoitteen kautta eli resurssi on saatavilla ja muunneltavissa asiakkaan ja palvelimen välillä. Vaikka REST-arkkitehtuurityylin mukaisesti jokaisen resurssin osoite on uniikki, voivat eri pyynnöt palveluille palauttaa saman resurssin. Web-sovelluksien kommunikointiprotokollana käytetään http-protokollaa, joten voimme käsitellä asiakas-palvelin-pyynnöissä mitä tahansa tietomuotoa. (Sandoval 2009, 26.)

Resurssien esittämisellä tarkoitetaan sitä, missä muodossa data kuljetetaan tai näytetään asiakaspuolen ja palvelimen välillä. Resurssin esittäminen on oikean datan väliaikainen tila, jonka esittämisen muoto on kuvattu datan pyynnön yhteydessä. Samaa dataa voidaan esittää monessa eri muodossa, kuten kuvana tai JSON-tietomuotona, mutta datan on tultava samasta URI-osoitteesta. Resurssien esittämisessä on olennaista, onko sitä tarkoitettu ihmisen luettavaksi vai ohjelmiston käsiteltäväksi. Esimerkiksi verkkosivu on resurssin esittämistä ihmiselle ymmärrettävässä muodossa, mutta on raskaampaa järjestelmille. Tietoa, joka ei ole suoranaisesti

tarkoitettu käyttäjälle esitettäväksi, voidaan esittää kevyemmässä muodossa, kuten XML-tietomuodossa. (Sandoval 2009, 27.)

3.2 **HTTP-metodit**

RESTful-tyylisissä web-sovelluksissa datan tilaa muutetaan ja käsitellään resurssien esittämisen kautta. Modernissa web-sovelluksien kehityksessä voidaan suunnittelua ja toteutusta rajoittaa, käyttämällä ennalta määritettyjä neljää käytäntöä resurssien käsittelyyn; CRUD eli ”Create, Retrieve, Update ja Delete”. (Sandoval 2009, 29.)

Yksinkertaisimmillaan RESTful-tyyliset web-palvelut ovat sovelluksia, jotka manipuloivat resurssien tilaa. CRUD-mallissa resurssien manipuloinnilla tarkoitetaan luomista, noutamista, päivittämistä ja poistamista. Jokaiseen CRUD-mallin kuvaamasta resurssin käsittelyn käyttötapauksesta löydetään samaa käyttötapausta palveleva HTTP-metodi. Näiden avulla voidaan luoda RESTful-tyylinen web-sovellus. (Sandoval 30.)

CRUD-mallin Create-käytäntöä resurssien luomiseen käytetään HTTP-metodia POST. POST-metodilla voidaan luoda uusia ainutlaatuisia resursseja sovelluksen tietokantaan. CRUD-mallin Retrieve-käytäntöä resurssien noutamiseen löytyy http-metodi GET. GET-metodilla voidaan noutaa resursseja, mikä johtaa usein jonkinlaiseen resurssien käsittelyyn, jossa resurssien esittäminen ja esittämistapa on tärkeää. CRUD-mallin Update-käytäntöön eli resurssien päivittämiseen löytyy http-metodi PUT. PUT-metodilla voidaan päivittää eli tehdä muutoksia, jos olemassa oleviin sovelluksen resursseihin. PUT-metodilla voidaan myös POST-metodin kaltaisesti luoda uusia resursseja. CRUD-mallin Delete-käytäntöön eli resurssien poistamiseen löytyy saman niminen http-metodi DELETE. DELETE-metodilla voidaan poistaa resursseja RESTful-sovelluksen tietokannasta. (Sandoval, 29-37.)

4 ZIMBRA

Zimbra on Synacor yrityksen kehittämä sähköposti, kalenteri ja yhteistyöskentely ratkaisu julkiselle ja yksityiselle sektorille. Zimbra ratkaisu tarjoaa työkaluja organisaatioiden sisäiseen viestintään sähköposti-, tiedostonjako- ja kalenterijärjestelmillään. Järjestelmät ovat saatavilla työpöytäversioina tai selainpohjaisina versioina ja toimivat useilla eri alustoilla ja laitteilla. Zimbra ratkaisusta on kaksi versiota: ilmainen avoimen lähdekoodin Open Source -versio ja maksullinen kaupallisen lähdekoodin Network Edition -versio. (Synacor n.d.a.)

4.1 Zimbra REST-ohjelmointirajapinta

Zimbra järjestelmiin on kehitetty REST-ohjelmointirajapinta, jonka avulla järjestelmien resursseihin voidaan päästä käsiksi ohjelmistokehityksessä ja ulkoisten järjestelmien integroinnissa. RESTful-sovellukset tarjoavat pääsyn järjestelmien resursseihin URL-polkujen kautta, käyttäen HTTP-metodeja. (Synacor n.d.b.)

Zimbra REST-ohjelmointirajapinta käyttää vain kahta HTTP-metodia rajapintametodeissaan: GET-metodia resurssien lukemiseen ja POST-metodia resurssien kirjoittamiseen ja muokkaamiseen. Zimbra REST-ohjelmointirajapinta tarjoaa metodeja sähköposti-, osoitekirja- ja kalenterijärjestelmiin. (Synacor n.d.b.)

4.2 Zimbra kalenterijärjestelmän REST-ohjelmointirajapinta metodit

Zimbra REST-ohjelmointirajapinta tarjoaa kolme metodia kalenterijärjestelmäänsä. Zimbra REST-ohjelmointirajapintaan tehtävät pyynnöt muodostetaan ohjelmakoodin 1 mukaisella rakenteella, jossa määritetään pyyntöjä tehtävän Zimbra-järjestelmän palvelimen protokolla ja osoitetietoja, halutun tietojen omistajan käyttäjätunnus ja mitä tietoa haetaan eli esimerkiksi sähköposteja tai kalenteria. Jokaiseen Zimbra REST-ohjelmointirajapinnan metodin käyttöön voidaan myös lisätä pyyntöjä tarkentavia parametrejä. Kaikki Zimbra kalenterin REST-ohjelmointirajapinnan metodit vaativat käyttäjän autentikoinnin. (Synacor n.d.b.)

```
{protocol}://{host}:{port}/home/{user}/{object}?{params}
```

Ohjelmakoodi 1. Zimbra REST-ohjelmointirajapinnan metodien URI-osoitteen rakenne.

Seuraavaksi käydään läpi lyhyesti, kuinka Zimbra REST-ohjelmointirajapinnan tarjoamia metodeja kalenteritietojen noutamiseen ja luomiseen.

Get Calendar -metodi palauttaa haetun kalenterin tapaamiset ja ajanvaraukset. Oletuksena käytetään järjestelmän oletuskalenteria, mutta käyttäjämääritettyjä kalentereita voidaan myös käyttää. Metodi käyttää HTTP-metodia GET resurssien hakemiseen. Metodi tukee useita eri tietomuotoja resurssien formatointiin kuten: .ics, json, xml, rss ja html. .ICS-tiedostomuoto on tekstitiedosto ja sisältää kalenteriin liittyviä tietoja ja yksityiskohtaisia tietoja kalenterissa olevista kalenteritapahtumista kuten: tapahtuman aikavälin, tapahtuman otsikon, kuvauksen ja tapahtuman tekijän tiedot. Metodiin käyttöön voidaan lisätä pyyntöä määrittäviä parametrejä kuten ohjelmakoodin 2 mukaisesti start- ja end-parametrit. Nämä parametrit määrittävät miltä aikaväliltä kalenteritietoja halutaan. Parametreissä määritetyt ajat tulee olla muodoltaan millisekunteina. Tämän metodin käyttämiseen täytyy pyynnössä määrittää fmt-parametrin arvoksi .ics ohjelmakoodin 2 mukaisesti. (Synacor n.d.c.)

```
http://localhost:7070/home/kayttaja/Calendar?fmt=ics&start=15147985170378end=1514799117037
```

Ohjelmakoodi 2. Get Calendar -metodin pyynnön rakenne.

Get FreeBusy -metodi palauttaa haussa määritetyn kalenterin free/busy tiedon ifb-tiedostomuodossa. IFB-tiedostomuoto on tekstitiedosto ja sisältää tietoja kalenterin varaustilanteista eli milloin kalenteri on FREE - vapaana ja milloin BUSY - varattuna. Tämän metodin käyttämiseen täytyy pyynnössä määrittää fmt-parametrin arvoksi ifb ohjelmakoodin 3 mukaisesti. Get FreeBusy -metodiin voidaan myös Get Calendar -metodin kaltaisesti lisätä aikaparametrit start ja end ohjelmakoodin 2 mukaisesti määrittämään pyynnölle haluttu aikaväli. Metodi käyttää HTTP-metodia GET resurssien hakemiseen. (Synacor n.d.d.)

```
http://localhost:7070/home/kayttaja/?fmt=ifb
```

Ohjelmakoodi 3. Get FreeBusy -metodin pyynnön rakenne.

Import Appointments -metodi avulla .ics-muodossa olevaa dataa voidaan tuoda kalenteriin. Kalentereissa olevat ajanvaraukset ja tapaamiset ovat usein .ics-muodossa olevaa dataa. Metodin avulla voidaan siirtää, päivittää tai luoda kalenterimerkintöjä. Metodi käyttää HTTP-metodia POST resurssien kirjoittamiseen. (Synacor n.d.e.)

5 CONFLUENCE PLUGIN

Confluence plugin järjestelmä tarjoaa mahdollisuuden asiakkaille, käyttäjille ja kehittäjille räätälöidä ja laajentaa Confluence-järjestelmää plugineilla eli lisäosilla. (Atlassian n.d.c) Confluence plugin järjestelmän avulla käyttäjät voivat lisätä Confluence-järjestelmään uusia makroja, tapahtumankuuntelijoita tai jopa aivan uusia ominaisuuksia. (Atlassian n.d.d)

Yksinkertaisimmillaan pluginin rakenne koostuu kooditiedostoista, resursseista ja Confluence-järjestelmälle ainutlaatuisesta konfigurointitiedostosta. Plugien käyttöönoton toteuttavat organisaation käyttäjät, joilla on Confluence-järjestelmän hallinnointioikeudet. Kuka tahansa kehittäjä voi tuottaa ja kehittää plugineja Confluence-järjestelmään, joko omalle alustalleen tai jakaa myös muille Confluence-järjestelmän käyttäjille. Osa Confluence-järjestelmän oletusominaisuuksista on toteutettu Confluence plugin järjestelmää käyttäen, vaikka ne ovatkin osa Confluencen ydintoiminnallisuutta. (Atlassian n.d.c.)

Jokainen Confluence plugin koostuu yhdestä tai useammasta plugin moduulista. Yksittäisellä pluginilla saattaa olla useampi toiminnallisuus, ja plugin moduulit ovat aina yhden toiminnallisuuden ilmentymä. Esimerkiksi, Confluence-järjestelmän ulkoasuteema-plugin koostuu useasta plugin moduulista: yhdestä ulkoasun väriteeman kuvaavasta plugin moduulista ja useammasta sivujen asetelmia kuvaavasta plugin moduulista. Tämän kaltainen plugin koostuu plugin moduuleista, jotka ovat riippuvaisia toistensa toiminnallisuuksista. (Atlassian n.d.c.)

Confluence-järjestelmän pluginit koostuvat enimmillään kolmesta erilaisesta komponentista: Plugin Descriptor -tiedostosta, Java-luokkatiedostoista (kooditiedostoista) ja resursseista. Kun nämä komponentit pakataan yhteen JAR-tiedostoon, saadaan Confluence plugin. (Atlassian n.d.d.)

5.1 JAR-tiedosto

JAR eli Java Archive, on tiedostomuoto, jonka avulla voidaan pakata useampi tiedosto yhteen tiedostoon. Tyypillisesti JAR-tiedosto koostuu: Java-tiedostoista (kooditiedostoista) ja resurssitiedostoista, joista kyseinen Java-sovellus on riippuvainen. Pakkaamalla Java-sovelluksen JAR-tiedostomuotoon on useita hyötyjä, ja on Java-koodin julkaisemisen yleisin tapa. JAR-tiedostomuotoon pakkaamalla, pienennät radikaalisti sovelluksen kokoa, teet sovelluksesta helpommin jaettavan, lyhennät sovelluksen latausaikaa ja voit muuttaa sovelluksesi lisäosaksi muiden käytettäväksi. (Oracle, n.d.a.)

JAR-tiedostot pakataan ZIP-tiedostomuotoon. Tämä tarkoittaa, että JAR-tiedostoja voidaan purkaa ja arkistoida. Tiedostoa voidaan pakata JAR-muotoon monella eri tavalla. Java Development Kit (JDK), tarjoaa Java

Archive Tool -komentokehotetyökalun, jonka avulla voit pakata tiedostoja JAR-muotoon komentokehotteen kautta. Useimmat Java-ohjelmointiympäristöt voivat pakata projektin JAR-tiedostomuotoon. (Oracle, n.d.b.)

5.2 Plugin Descriptor

Plugin Descriptor on ainutlaatuinen XML-tiedosto nimeltään atlassian-plugin.xml. Jokainen Atlassianin järjestelmän plugin tarvitsee Plugin Descriptor -tiedoston, jotta se voidaan asentaa ja ottaa käyttöön Atlassianin järjestelmissä. Plugin Descriptor -tiedosto kuvaa välttämättömiä, ja vaihtoehtoisia, tietoja pluginista. Plugin Descriptor -tiedosto myös kuvaa pluginin sisältämät plugin moduulit ja resurssit. Plugin Descriptor -tiedosto sijaitsee pluginin projektikansion resources-kansiossa. (Atlassian, n.d.f.)

Ohjelmakoodi 4 esittää yksinkertaisen esimerkin atlassian-plugin.xml-tiedoston rakenteesta ja seuraavaksi käyn läpi tarkemmin tämän tiedoston sisältöä.

atlassian-plugin.xml

```
<atlassian-plugin key="com.atlassian.confluence.plugins.example"
                  name="My Plugin" plugins-version="2">

<plugin-info>
    <description>
      A sample plugin for demonstrating the file format.
    </description>
    <version>1.0</version>
    <application-version min="1.3" max="1.3"/>
    <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com/"/>
    <param name="configure.url">/admin/plugins/example/configurePlugin.action</param>
    <bundle-instructions>
      <Export-Package>my.external.pkg</Export-Package>
      <Import-Package>com.mylibrary,*;resolution:=optional</Import-Package>
    </bundle-instructions>
  </plugin-info>

  <examplemodule key="module1" name="Example Module"
                 class="com.atlassian.confluence.plugins.example.ExampleModule">
    <description>An example module</description>
  </examplemodule>
</atlassian-plugin>
```

Ohjelmakoodi 4. atlassian-plugin.xml-tiedosto esimerkki.

Atlassian-plugin -elementti on atlassian-plugin.xml -tiedoston juurielementti. Tämä elementti sisältää key-attribuutin, jonka tulee sisältää ainutlaatuinen merkkijono, jonka avulla plugin on tunnistettavissa. Name-attribuutti sisältää pluginin nimen, joka esiintyy muun muassa Confluence-järjestelmän käyttöliittymän menuissa. (Atlassian, n.d.f.)

Plugin-info -elementti sisältää useita pluginia kuvaavia elementtejä. Description-elementti sisältää luettavan kuvauksen pluginista. Kuvaus on

nähtävissä Confluence-järjestelmän pluginin asennuksien ja käyttöönottojen valintanäkymissä. Version-elementti sisältää tämän pluginin nykyisen version. Application-version -elementti sisältää minimi ja maksimi määritykset siitä, mitkä järjestelmäversiot ovat yhteensopivia tämän pluginin kanssa. (Atlassian, n.d.f.)

Seuraavat plugin-info -elementin sisällä olevat elementit ovat vaihtoehtoisia tietoja ja määrityksiä. Vendor-elementti kuvaa name-attribuutissa pluginin tarjoajan nimen ja url-attribuutissa tarjoajan sivuston. Param-elementti on sisältää arbitraarisia parametrejä pluginille. Param-elementin name-attribuutti kuvaa parametrin nimen ja sisältö parametrin arvon. Param-elementin voi sisällyttää myös usean muun atlassian-plugin.xml tiedoston elementin sisälle. Bundle-instructions -elementin avulla pluginille voidaan määrittää yksityiskohtaisia riippuvuuksia. (Atlassian, n.d.f.)

Plugin-info -elementin jälkeen kuvataan pluginin sisältämät plugin moduuli elementit. Esimerkissä oleva examplemodule-elementti ei ole aito plugin moduulityyppi, mutta sisältää tyypillisiä attribuutteja, joita löytyy useammista plugin moduuleista. Examplemodule-elementin key-attribuutti sisältää ainutlaatuisen merkkijonon, jonka avulla moduulin voidaan viitata. Name-attribuutti sisältää luettavan nimen moduulille ja class-attribuutti sisältää plugin projektikansion polun moduulin Java-luokkatiedostoon. Vaihtoehtoisesti plugin-moduuleille voidaan lisätä description-elementti, joka sisältää kuvauksen plugin-moduulista. (Atlassian, n.d.f.)

5.3 **Java-luokkatiedostot ja pluginin resurssit**

Confluence-järjestelmään JAR-tiedostona asennettu plugin kykenee käyttämään kaikkia JAR-tiedostoon pakattuja Java-luokkatiedostoja. Useimmat plugin-moduulit ovat Java-luokkatiedostoja ja ne voivat käyttää muita luokkia toiminnallisuudessaan. Confluence pluginit voivat sisältää niin monta luokkatiedostoa kuin pluginin toiminnallisuus vaatii. (Atlassian, n.d.d.)

Confluence pluginiin voi määrittää ladattavia resursseja, jotka ovat välttämättömiä pluginin toiminnallisuuteen. Confluence plugin resurssi on melkein mikä tahansa resurssi tai komponentti joka ei ole Java-luokkatiedosto. Tyypillisiä pluginin resursseja ovat muun muassa: CSS-tyylitiedostot, JavaScript-tiedostot tai -kirjastot, Velocity järjestelmän kaavaintiedostot HTML-luomiseen ja kuvat. Confluence pluginin ladattavat resurssit voidaan määrittää yleisesti osana pluginia tai osana resurssikokonaisuutta. (Atlassian, n.d.g.)

Ladattavat resurssit merkitään Plugin Descriptor -tiedostoon ohjelmakoodin 5 esimerkin mukaisesti resource-elementteinä.

```
<resource type="velocity" name="template"
          location="com/example/plugin/template.vm"/>

<resource type="download" name="style.css"
          location="com/example/plugin/style.css">
   <property key="content-type" value="text/css"/>
</resource>
```

Ohjelmakoodi 5. resource-moduuli esimerkki.

Resource-elementit sisältävät kolme eri attribuuttia. Type-attribuutti sisältää resurssin tyypin ja kertoo moduulille, miten resurssia tulee käyttää. Name-attribuutti sisältää käyttävän resurssin tiedostonimen. Ohjelmakoodin 5 alemmassa resource-elementissä huomataan, että name-attribuutti sisältää resurssin tiedostotyypin .css. Tiedostotyyppi tulee olla name-attribuutissa, jos resurssi on CSS- tai JavaScript-tiedosto. Location-attribuutti sisältää resurssin projektikansion polun. Resource-elementit voivat myös sisältää property-elementtejä. Property-elementit sisältävät arbitraarisia avain-arvo-pareja. (Atlassian, n.d.g.)

Jos plugin vaatii useita staattisia resursseja kuten CSS- ja JavaScript -tiedostoja, käytä Web Resource -moduulia. Web Resource -moduuli lisää resurssit pluginia käyttävän HTML-sivun header-elementtiin. Web Resource -moduulia käytetään ohjelmakoodin 6 mukaisesti, lisäämällä web-resource -elementti Plugin Descriptor -tiedostoon. (Atlassian, n.d.h.)

```
<web-resource key="scriptaculous" name="Scriptaculous" >
    <resource type="download" name="scriptaculous.js"
              location="includes/js/effects/scriptaculous.js" />
    <resource type="download" name="effects.js"
              location="includes/js/effects/effects.js" />
</web-resource>
```

Ohjelmakoodi 6. web-resource -moduuli esimerkki.

Web-resource -elementti koostuu tässä kappaleessa aiemmin mainituista resource-elementeistä. Ohjelmakoodin 6 web-resource -elementti sisältää kaksi attribuuttia. Key-attribuutti sisältää yksilöivän merkkijonon, johon viittaamalla resurssia voidaan käyttää muun muassa pluginin Java-luokkatiedostoissa. Name-attribuutti sisältää resurssia kuvaavan nimen. (Atlassian, n.d.h.)

5.4 Makromoduuli ja Velocity Template Engine

Makro on lisäosa, joka voidaan lisätä Confluence-järjestelmän sivulle sivun luomisen tai muokkaamisen yhteydessä makrohallinnointinäkymän avulla. Makromoduuli määritetään Plugin Descriptor -tiedostossa ja sisältää aina

Java-luokkatiedoston, jossa määritetään mitä makro tuo Confluence sivulle. Makron Java-luokkatiedosto ajetaan makron sivulle lisäyksen yhteydessä ja tulostaa sivulle tiedostossa määritetyn html-tietomuodossa olevan datan. (Atlassian, n.d.i.)

Velocity Template Engine eli karkeasti suomennettuna Velocity kaavainjärjestelmä, on Java-pohjainen kaavainjärjestelmä. Sillä käyttäjät voivat käyttää yksinkertaista kaavainohjelmointikieltä, joka muuntaa Java-ohjelmointikoodissa määritettyjä tietoja html-tietomuotoon. Java-ohjelmointikoodissa tietoja määritetään ja lisätään Context-olioon, joka on periaatteessa yksinkertainen avainarvoparitaulukko. Velocity kaavainjärjestelmän toiminnallisuus perustuu MVC-ohjelmointimalliin, Context-olion alustaminen ja määrittäminen tehdään Java-ohjelmointikooditiedostoissa ja html-kaavain on Velocity kaavainjärjestelmän ainutlaatuisessa .vm-tiedostossa. (The Apache Software Foundation. n.d.b.)

Kuva 1. Velocity Template Engine käytön esimerkki ja makromoduulin kanssa.

5.5 Servlet-moduuli

Servlet on Java ohjelmointikielen luokka, jonka avulla palvelin puolen toimintoja ja palveluja voidaan laajentaa asiakaspuolelle. Servletin toiminnot ja palvelut ovat saatavilla request-response -ohjelmointimallin avulla. Tämä tarkoittaa, että servlet käyttöliittymä sijaitsee palvelimella isännöidyssä URI-osoitteessa, johon voidaan tehdä GET ja POST tyyppisiä HTTP-pyyntöjä. Servlet suorittaa pyynnöt doGet- ja doPost-metodeilla ja voi tarvittaessa lähettää viestin pyynnön lähettäneeseen lähteeseen Response-olion avulla. (Oracle, n.d.c.)

Servlet-moduulin avulla Confluence-järjestelmään voidaan asentaa tyypillinen servlet palvelu. Servlet-moduuli määritetään Plugin Descriptor -tiedostossa. Servletin URI-osoite koostuu Confluence-järjestelmän palvelimen osoitteesta, ”plugins/servlet/” -osoitteen jatkosta ja Plugin Descriptor -tiedoston servlet-elementissä määritetystä url-pattern -elementin sisältämästä arvosta. (Atlassian, n.d.j.)

```
<servlet name="Esimerkki Servlet" key="esimerkkiServlet"
         class="com.esimerkki.Servlet">
    <description>Servletin kuvaus</description>
    <url-pattern>/URIosoite</url-pattern>
</servlet>
```

Ohjelmakoodi 7. Servlet-moduuli esimerkki.

6 ATLASSIAN SDK:N KÄYTTÄMINEN

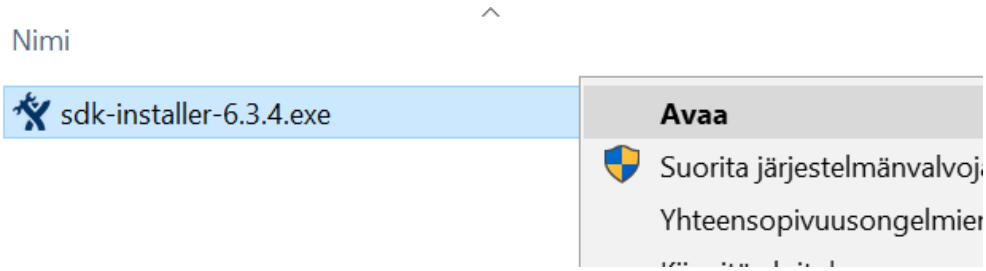
6.1 Atlassian SDK:n asennus Windows-käyttöjärjestelmään

Atlassian SDK on asennettavissa Windows, Mac ja Linux käyttöjärjestelmiin. Tämä ohjeistus keskittyy kehittäjäpaketin asentamiseen Windows-käyttöjärjestelmälle.

Ennen Atlassian SDK:n asentamista täytyy käyttöjärjestelmään asentaa Java SDK ja määrittää välttämättömät ympäristömuuttujat, jotta voidaan käyttää Java SDK:n työkaluja käyttöjärjestelmässä. Tämä ohjeistus Atlassian SDK:n asentamiseen Windows-käyttöjärjestelmään olettaa, että edellä mainitut asennukset ja määritykset on suoritettu.

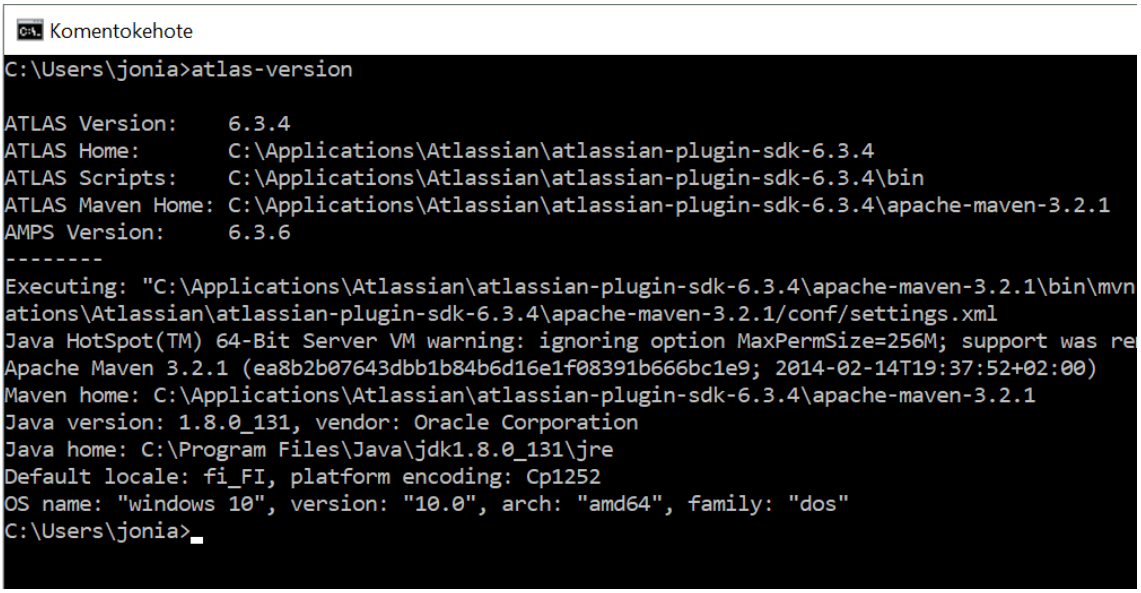
Atlassian SDK-kehittäjäpaketin asennus on erittäin suoraviivainen ja helppo prosessi. Ladataan uusin versio Atlassian SDK-kehittäjäpaketista osoitteesta: https://marketplace.atlassian.com/download/plugins/atlassian-plugin-sdk-windows

Paikannetaan ladattu Atlassian SDK asennustiedosto ja avataan tiedosto joko kaksoisklikkaamalla pikakuvaketta tai kuvan 2 mukaisesti, oikealla hiirenpainikkeella klikkaamalla pikakuvaketta ja valitsemalla ”Avaa”-toiminnon.



Kuva 2. Atlassian SDK asennustiedoston avaaminen.

Kun asennus on suoritettu, voidaan varmistaa vielä asennuksen onnistuminen komentokehotteen kautta. Avataan komentokehoteikkuna ja syötetään tekstikenttään komento ”atlas-version”, kuvan 3 mukaisesti. Jos asennus on suoritettu onnistuneesti, saadaan komentokehotteeseen kuvan 3 mukaista tietoa asennetusta Atlassian SDK versiosta.



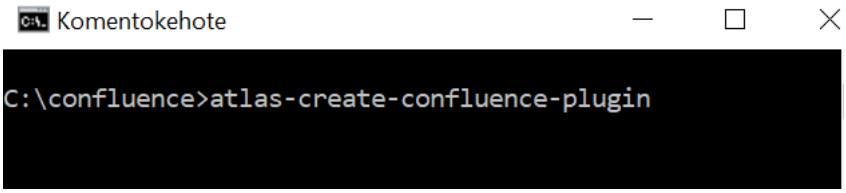
Kuva 3. Atlassian SDK versio tietoja komentokehotteessa.

Asennus on onnistunut. Asennuksen jälkeen työympäristö on valmis Atlassianin järjestelmien pluginien kehittämiseen.

6.2 Hello World - Confluence pluginin luominen

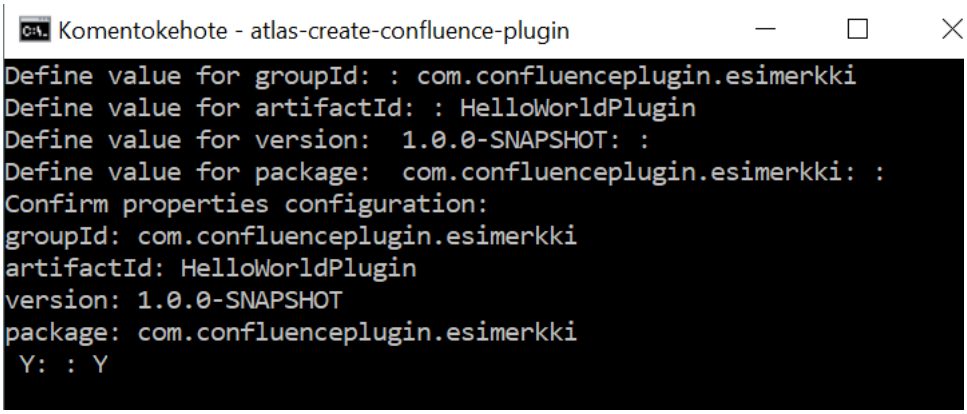
Tässä kappaleessa käydään läpi, kuinka luodaan erittäin yksinkertainen Confluence-järjestelmän plugin. Luodaan xhtml-macro -moduulia käyttävä ”Hello World” plugin. ”Hello World” plugin on Confluence-järjestelmän makro. Lisäämällä pluginissa luodun makron Confluence-järjetelmän sivulle, tuo makro sivulle ”Hello World” tekstin käärittynä <h1> html-elementtiin. Confluence plugin voidaan luoda Atlassian SDK -kehittäjäpaketin komentokehotetyökaluilla.

Syötetään komentokehotteeseen komento ”atlas-create-confluence-plugin” kuvan 4 mukaisesti. Tämä luo Confluence-järjestelmään yhteensopivan plugin projektin, jossa ei ole vielä mitään toiminnallisuutta.



Kuva 4. ”atlas-create-confluence-plugin” komento.

Komennon syötettyä, pitää plugin projektille antaa muutamia arvoja kuten pluginin nimi ja projektikansion polku. Syötetään kuvan 5 mukaiset arvot ”groupId”- ja ”artifactId” tiedoille. Ohitetaan version ja paketin arvojen määrittäminen ja hyväksytään plugin projektin määritykset syöttämällä ”Y”-merkki.



Kuva 5. ”atlas-create-confluence-plugin” komennolla luodun projektin määritykset.

Nyt ollaan luotu valmis plugin projektipohja kansioon, jossa syötettiin ”atlas-create-confluence-plugin” komento. Plugin projektipohja pluginille sisältää välttämättömiä määritys tiedostoja ja tyhjiä vaihtoehtoisia tiedostopohjia kuten: JavaScript-, CSS- ja properties-tiedostot.

Jotta saadaan toimiva ”Hello World” plugin, täytyy lisätä xhtml-macro -moduuli plugin atlassian-plugin.xml tiedostoon ja luoda siihen viittaavaa Java-luokkatiedosto. Xhtml-macro -moduuli lisätään atlassian-plugin.xml tiedostoon web-resource -moduulin jälkeen ohjelmakoodin 8 mukaisesti.

Xhtml-macro -moduuli sisältää tietoja makrosta kuten: nimen, yksilöidyn tunnuksen, toiminnallisuuden kuvauksen ja mahdollisia parametrejä. Makron parametrit voivat olla, joko pluginissa ennalta määritettyjä tai käyttäjän syöttämiä makron käyttöönoton yhteydessä. Tärkeimpiä tietoja on makron ”class”-attribuutti, joka osoittaa makron suorittamaan Java-luokkatiedostoon.

atlassian-plugin.xml

```
<atlassian-plugin key="${atlassian.plugin.key}" name="${project.name}" plugins-version="2">
    <plugin-info>
        <description>${project.description}</description>
        <version>${project.version}</version>
        <vendor name="${project.organization.name}" url="${project.organization.url}" />
        <param name="plugin-icon">images/pluginIcon.png</param>
        <param name="plugin-logo">images/pluginLogo.png</param>
    </plugin-info>

    <!-- add our i18n resource -->
    <resource type="i18n" name="i18n" location="HelloWorldPlugin"/>

    <!-- add our web resources -->
    <web-resource key="HelloWorldPlugin-resources" name="HelloWorldPlugin Web Resources">
        <dependency>com.atlassian.auiplugin:ajs</dependency>

        <resource type="download" name="HelloWorldPlugin.css" location="/css/HelloWorldPlugin.css"/>
        <resource type="download" name="HelloWorldPlugin.js" location="/js/HelloWorldPlugin.js"/>
        <resource type="download" name="images/" location="/images"/>

        <context>HelloWorldPlugin</context>
    </web-resource>
    <xhtml-macro name="helloworld" class="com.confluenceplugin.esimerkki.helloworld"
                 key='helloworld-macro'>
        <description>Hello World Plugin</description>
        <category name="formatting"/>
        <parameters/>
    </xhtml-macro>
</atlassian-plugin>
```

Ohjelmakoodi 8. Xhtml-macro moduulin lisääminen atlassian-plugin.xml tiedostoon.

Seuraavaksi luodaan helloworld.java Java-luokkatiedoston com.confluenceplugin.esimerkki kansioon.

Helloworld.java tiedoston sisältö koostuu ohjelmakoodin 9 mukaisesta koodista. Helloword Java-luokka perii Atlassianin Macro luokan, jonka takia määritetään muutamat oletusmetodit luokkaan, joista tärkein on execute-metodi. Execute-metodi palauttaa merkkijonon, joka sijoitetaan Confluence-sivulle, jonne Confluence makro on laitettu sivun luonnin tai muokkauksen aikana. Ohjelmakoodin 9 execute-metodi palauttaa merkkijonon, joka sisältää <h1> html-elementtiin käärittynä merkkijonon ”Hello World!”.

helloworld.java

```
package com.confluenceplugin.esimerkki;

import com.atlassian.confluence.content.render.xhtml.ConversionContext;
import com.atlassian.confluence.macro.Macro;
import com.atlassian.confluence.macro.MacroExecutionException;

import java.util.Map;

public class helloworld implements Macro {

    public String execute(Map<String, String> map, String s,
                          ConversionContext conversionContext)
                          throws MacroExecutionException {
        return "<h1>Hello World!</h1>";
    }

    public BodyType getBodyType() { return BodyType.NONE; }

    public OutputType getOutputType() { return OutputType.BLOCK; }
}
```

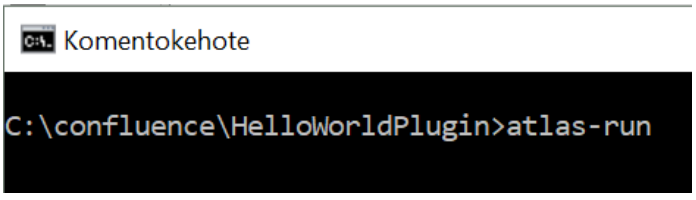
Ohjelmakoodi 9. Helloworld.java tiedoston sisältö.

”Hello World” plugin on valmis. Seuraavassa kappaleessa käydään läpi, kuinka plugin asennetaan ja kuinka makro voidaan ottaa käyttöön Confluence-järjestelmän sivulla.

6.3 Confluence Demonstration Space - plugin testausympäristö

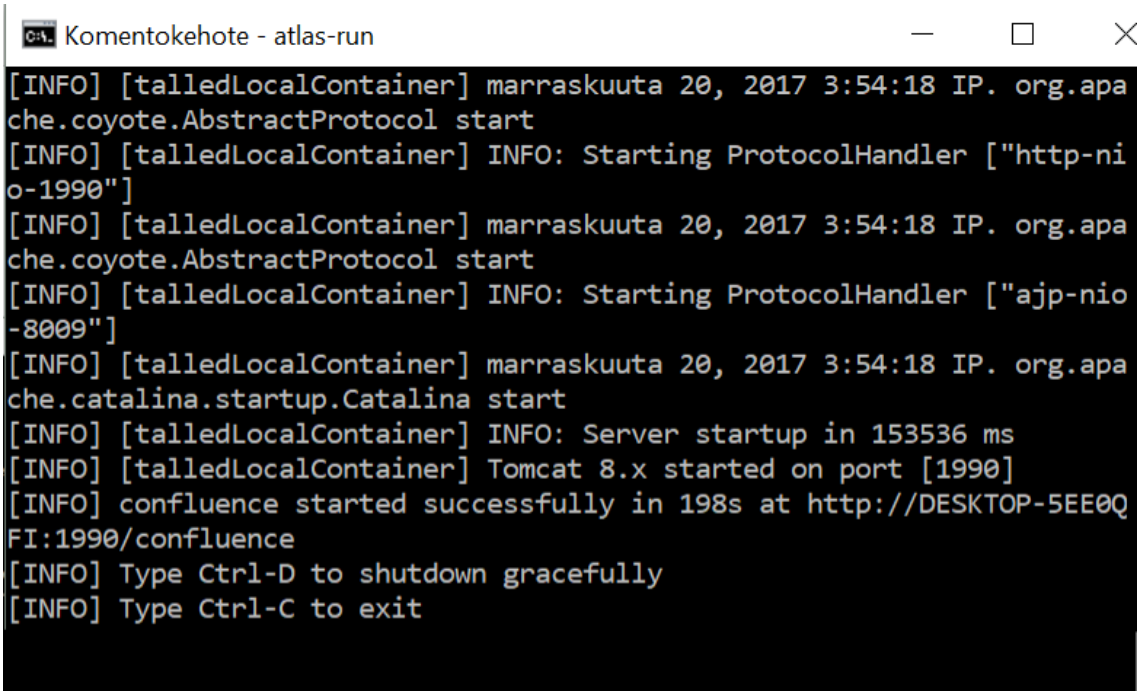
Atlassianin Confluence Demonstration Space on Confluence-järjestelmän demonstrointitila, jossa voidaan käyttää ja kokeilla useimpia Confluence-järjestelmän ominaisuuksia ja toiminnallisuuksia. Confluence Demonstration Space toimii myös Confluence plugin testausympäristönä.

Uutta kehitettävää pluginia voidaan testata demonstrointitilassa ajamalla plugin komentokehotteen kautta käyttämällä Atlassian SDK -kehittäjäpaketin komentokehotetyökaluja. Siirrytään kehitettävän pluginin juurikansioon komentokehoteikkunassa ja syötetään komento ”atlas-run” kuvan 6 mukaisesti.



Kuva 6. ”atlas-run” komento.

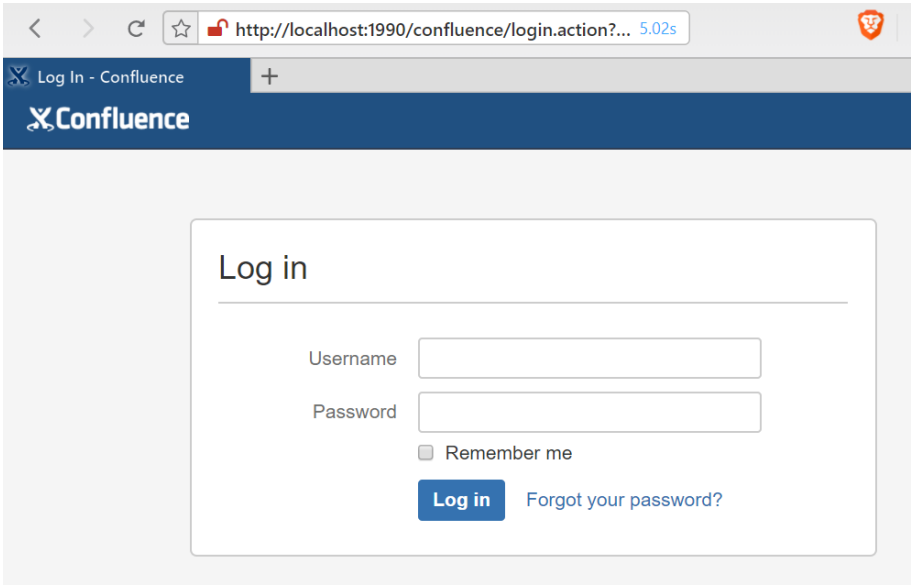
Komento ”atlas-run” lataa tarvittavat riippuvuudet Confluence Demonstration Space ajamiseen, asentaa pluginin järjestelmään ja suorittaa Confluence-järjestelmän ajamisen koneen paikallisella palvelimella. ”atlas-run” komennon suorittaminen voi kestää jopa muutaman minuutin. Kun Confluence-järjestelmän ajaminen on onnistunut, nähdään komentokehotteessa kuvan 7 mukaista tietoa.



Kuva 7. ”atlas-run” komento on suoritettu onnistuneesti.

Confluence Demonstration Space ajetaan tietokoneen paikallisella palvelimella. Tomcat-palvelin ajaa demonstrointitilaa portissa 1990, joten Confluencen järjestelmään päästään selaimella osoitteessa localhost:1990/confluence, kuten komentokehote kertoo kuvassa 7.

Avataan selain ja siirrytään osoitteeseen http://localhost:1990/confluence. Osoitteesta löytyy Confluence-järjestelmän kuvan 8 kaltainen sivu sisäänkirjautumiseen.



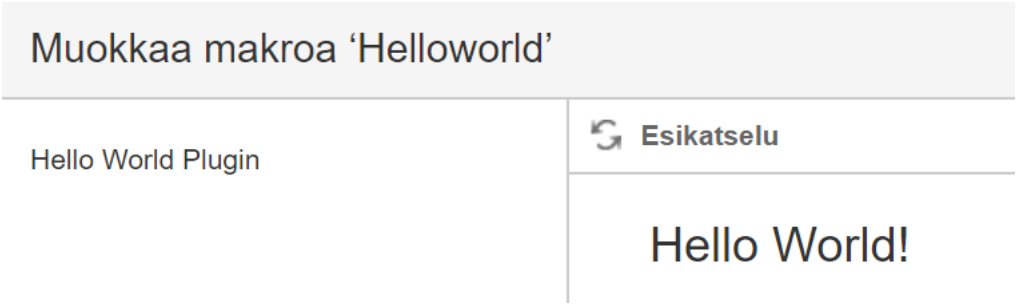
Kuva 8. Confluence demonstrointitilan sisäänkirjautumissivu.

Confluence-järjestelmän demonstrointitila vaatii sisäänkirjautumisen. Demonstrointitilan sisäänkirjautumiseen käytetään käyttäjätunnusta ”admin” ja salasanaa ”admin”.

Jotta voidaan testata kehitettyä ”Hello World” makroa, täytyy luoda uusi sivu Confluence Demonstration Space -tilaan. Luodaan uusi sivu ja avataan sivun muokkaustilassa makrohallinnointinäkymä. Makrohallinnointinäkymässä löydetään kuvan 9 mukainen ”Hello World” makro. Huomataan, että makron thumbnail sisältää xhtml-macro -moduulissa määritetyn name-attribuutin nimenä ja description-elementtiin annnetun makron kuvauksen.

Kuva 9. Hello World makro.

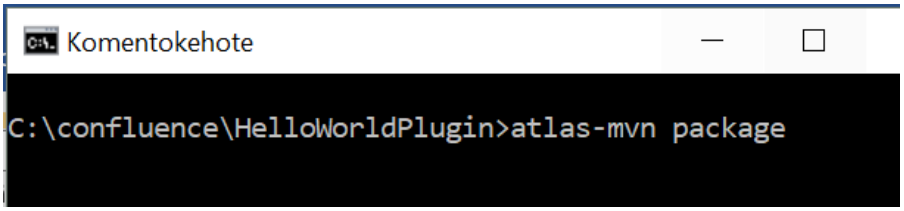
Valitsemalla makron, avautuu uusi näkymä, jossa voidaan esikatsella valittua makroa. Kuvan 10 esikatselunäkymässä näkyy ”Hello World!” merkkijono, joten kehitetty plugin toimii oikein.



Kuva 10. Hello World makron esikatselu.

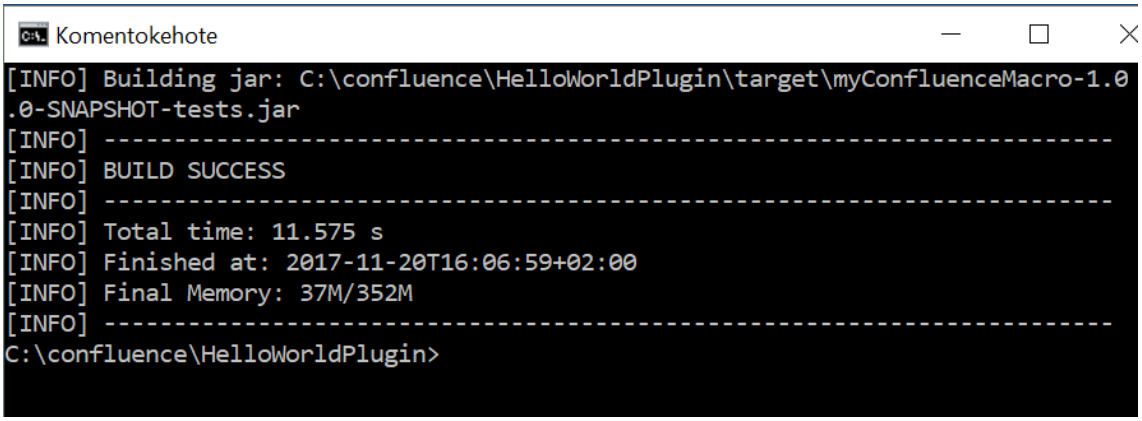
Kuten aikaisemmin mainittiin, Confluence-järjestelmän ajaminen voi viedä paljon aikaa. Onneksi voidaan päivittää pluginiin tehtyjä muutoksia Confluence-järjestelmän ajon aikana, ilman ajon uudelleen käynnistämistä. Tämä on keskeinen osa pluginin kehitystyötä.

Jotta voidaan syöttää komento ajonaikaiseen päivitykseen, pitää avata uusi komentokehoteikkuna ja siirtyä pluginin juurikansioon, jossa Confluence-järjestelmän ajo on käynnissä aikaisemmin avatussa komentokehoteikkunassa. Käyttämällä kuvan 11 mukaista komentoa ”atlas-mvn package”, voidaan päivittää Confluence-järjestelmään asennettu plugin.



Kuva 11. ”atlas-mvn package” komento.

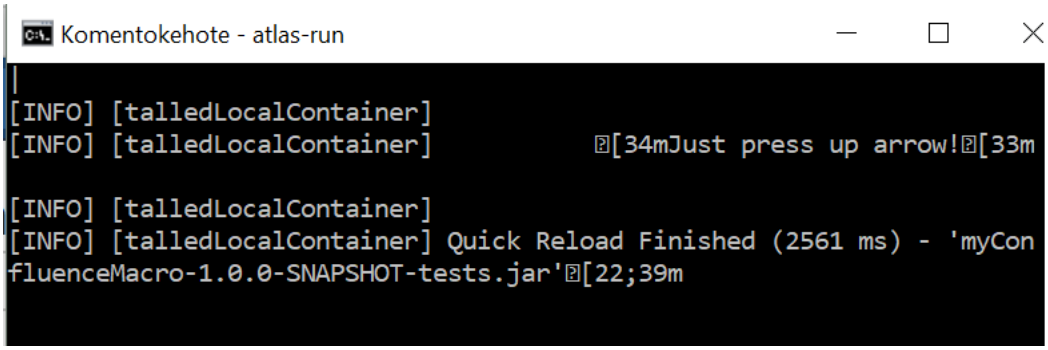
”atlas-mvn package” uudelleen pakkaa ja asentaa kehitetyn pluginin ilman Confluence-järjestelmän demonstrointitilan uudelleen käynnistämistä. Pluginin uudelleen pakkaus on onnistunut, kun nähdään kuvan 12 mukainen viesti komentokehotteessa.



Kuva 12. Pluginin uudelleen asennus on onnistunut ”atlas-mvn package” komennolla.

Komentokehoteikkunassa, jossa ”atlas-run” toiminto on käynnissä, huomaataan ”atlas-mvn package” uudelleen pakkaamisen onnistuneen, kun

nähdään kuvan 13 mukainen viesti ”Quick Reload Finished”. Nyt voidaan testata, nähdäänkö pluginiin tehtyjä muutoksia Confluence-järjestelmässä.



Kuva 13. ”Quick Reload Finished” uudelleen asennus on onnistunut.

Confluence-järjestelmän ajaminen voidaan lopettaa komentokehotteen kautta. Annetaan komento CTRL-Z komentokehotteessa ja varmennetaan komento ENTER-painikkeella. Tämä komento lopettaa Confluence-järjestelmän ajamisen turvallisesti, mutta kestää jonkin aikaa. Tämä on paras tapa lopettaa järjestelmän ajaminen. Ajaminen voidaan lopettaa myös komennolla CTRL-C, mutta tämä komento saattaa jättää taustaohjelmia pyörimään, jotka voivat estää järjestelmän uudelleen ajamisen onnistuneesti.

7 PLUGININ SUUNNITTELU

Tässä kappaleessa käydään läpi asiakkaan pyytämän Confluence-järjestelmän pluginin käyttötapaus, toiminnallisuuteen määritetyt vaatimukset, käyttöliittymä, kehittämiseen valittu toteutustapa ja teknologiat. Kuvattujen asioiden pohjalta kykenin aloittamaan Confluence pluginin kehittämisen suunnittelun.

7.1 Käyttötapaus ja vaatimukset

Asiakkaan pyytämän pluginin käyttötapaus on yksinkertainen; asiakkaan työntekijä saa puhelun yrityksen asiakkaalta ja työntekijän pitäisi saada nopeasti tietoa, mihin neuvotteluhuoneeseen hän voi mennä keskustelemaan asiakkaan kanssa rauhassa. Plugin sijaitsee asiakkaan Confluence-järjestelmän etusivulla, joten työntekijä avaa Confluence etusivun selaimellaan, jossa neuvotteluhuoneiden varaustilannetta tarkkaileva plugin näyttää, mitkä toimiston neuvotteluhuoneet ovat vapaina juuri sillä hetkellä.

Kun Confluene-järjestelmän pluginista keskusteltiin opinnäytetyön asiakkaan kanssa, saatiin määriteltyä muutamia perusvaatimuksia pluginin toiminnallisuudelle. Pluginin tulee pystyä tarkkailemaan neuvotteluhuoneiden varaustilannetta minuutin välein. Pluginin tulee pystyä päivittämään käyttöliittymäänsä dynaamisesti eli käyttöliittymän päivitys ei vaadi sivun uudelleen latautumista. Pluginin tulee hakea kalenterienvaraustiedot käyttäen Zimbra kalenterijärjestelmän REST-ohjelmointirajapintaa. Asiakas ei määrittänyt, millä teknologioilla ja toteutustavoilla pluginin tulee kehittää, joten muutamaa erilaista toteutustapaa tutkitaan myöhemmin tässä kappaleessa.

7.2 Käyttöliittymä

Pluginin käyttöliittymä on yksinkertainen: se koostuu neliöistä, jotka esittävät toimiston neuvotteluhuoneita. Neliön väri perustuu neuvotteluhuoneen nimeen, esimerkiksi kuvan 14 sininen neliö on neuvotteluhuone Bluescreen ja vihreä neliö on neuvotteluhuone Greenhouse.

Kuva 14. Pluginin makrossa neuvotteluhuoneita esittävät värineliöt.

Käyttöliittymän lähtökohtainen idea on, kun neuvotteluhuone on vapaana, näkyy neuvotteluhuonetta kuvaava neliö värikkäänä. Toisin, kun neuvotteluhuone on varattuna, näkyy neuvotteluhuonetta kuvaava neliö

harmaansävyisenä. Esimerkkinä, kuvassa 15 neuvotteluhuone Bluescreen on havainnollistettu varattuna, koska se on väriltään harmaa.

Kuva 15. Makrossa neuvotteluhuone Bluescreen varattuna.

Toteutuksen aikana tuli esille asiakkaan kanssa pluginin käyttöliittymää parantavia ominaisuuksia. Asiakas haluaa nähdä lisätietoja neuvotteluhuoneen varauksesta. Haluttuja lisätietoja olisivat: neuvotteluhuoneen varaajan nimitiedot, huoneen varaustilanne ja varauksen aikatiedot eli mille aikavälille huone on varattu. Kuvan 16 mukaisesti, nämä tiedot näytettäisiin käyttäjälle tooltip-viestinä, kun käyttäjä siirtää hiirensä värineliön päälle.

Kuva 16. Varatun neuvotteluhuoneen varaustilanteen lisätiedot.

Pluginin kehityksen ohella törmättiin ongelmaan, että jos useampi neuvotteluhuone on varattuna eli useampi värineliö on harmaana, tulee vaikeuksia tietää mitkä huoneet ovat varattuina ilman lisätietojen tarkastelua. Tämä ongelma ratkaistiin kuvan 17 osoittamalla tavalla. Värineliöön lisättiin värikulma, joka säilyttää värineliön alkuperäisen värin silloin, kun värineliö on harmaana. Tällä tavoin voidaan yksinkertaisesti visuaalisesti viestiä, mikä huone on varattuna.

Kuva 17. Värikulma varattuna tilassa olevassa värinelilössä.

7.3 **Toteutustapa**

Confluence pluginin toteutustavan valinta alkoi tutustumisella Confluence plugin kehitystyökaluihin ja järjestelmän tarjoamiin ominaisuuksiin ja

plugin moduuleihin. Tämän projektin pluginille määritetyistä vaatimuksista dynaaminen käyttöliittymän päivitys vaikutti merkittävimmin toteutustavan valintaan. Päädyttiin toteuttamaan käyttöliittymän eli makron dynaaminen päivitys JavaScript-tiedostolla, joka lisätään pluginiin resurssina.

Seuraava merkittävä kysymys oli, kuinka hoidetaan pluginin taustatoiminnat eli millä tavoin tehdään pyyntöjä Zimbra kalenterin REST-ohjelmointirajapintaan, kuinka pyynnöt käsitellään ja lähetetään tiedot varaustilanteiden muutoksista pluginin käyttöliittymään. Ensimmäisenä lähestyttiin ja tutkittiin pluginin toteutusta puhtaasti JavaScriptillä. Kaikki aiemmin mainitut taustatoiminnot toteutettaisiin JavaScriptillä, mutta törmättiin ongelmaan aikaisin tässä toteutustavassa. JavaScriptillä voidaan tehdä http-pyyntöjä, mutta merkittäviä rajoituksia pyyntöjen tekemiseen tulee, kun halutaan tehdä http-pyyntöjä selaimella ajettavalla JavaScript koodilla. Selaimella ajettavan JavaScript koodilla tehtävien http-pyyntöjen tekemistä on rajoitettu lukuisista turvasyistä. Merkittävin rajoitus on, että pyyntöjä ei voida tehdä eri palvelimella oleviin resursseihin ilman palvelimella toteutettuja konfiguraatiota, jotka sallivat pyynnöt määritetyistä osoitteista. Tämä tarkoittaa, että makron JavaScript resursseista http-pyyntöjen tekeminen asiakkaan Zimbra kalenterin REST-ohjelmointirajapintaan, joka pyörii toisella palvelimella, ei ole sallittua.

Vaihtoehtoisena taustatoimintojen toteutustapana, jonka kanssa myös lähdettiin pluginia toteuttamaan, on lisätä pluginiin servlet-moduuli. Javalla toteutettu servlet hoitaa aiemmin mainitut pluginin palvelimen puoleiset vaaditut toiminnallisuudet. Palvelimella ajettava ohjelma voi tehdä http-pyyntöjä resurssien hakemiseen moitteettomasti, joten servlet kykenee tekemään tarvittavat http-pyynnöt. Pluginin käyttöliittymää päivittävä JavaScript tekee http-pyynnön pluginissa määritettyyn servletiin, joka sijaitsee samalla palvelimella, joten pyyntöjen tekeminen on sallittua. Kun pyyntö on käsitelty servletin puolella, lähettää servlet vastauksena tiedon JavaScriptiin neuvotteluhuoneiden varaustilanteista, joiden perusteella JavaScript päivittää makron käyttöliittymän.

8 PLUGININ TOTEUTUS

Tässä kappaleessa käydään läpi asiakkaalle kehittämän Confluence plugin kehitystyön vaiheet ja prosessit. Plugin kehitystyön kerrontaa lähestytään käytännön läheisesti. Jokainen tämän kappaleen osa-alue keskittyy pluginin keskeisen ominaisuuden tai toiminnallisuuden luomiseen.

Confluence pluginin toteutus alkoi uuden plugin projektin luomisella käyttäen Atlassian SDK -kehittäjäpaketin työkaluja. Toteutuksen työkaluina käytin Atlassian SDK -kehittäjäpaketin komentokehotetyökaluja, Atom koodieditoria ja selainta, jossa pystyttiin käyttämään Confluence Demonstration Space ympäristöä pluginin testaamiseen.

8.1 Makron luominen ja dynaaminen päivitys

Aloitetaan kehitystyö luomalla pluginin käyttöliittymän eli makron. Makron luominen toteutetaan lisäämällä Plugin Descriptor tiedostoon eli atlassian-plugin.xml tiedostoon, ohjelmakoodin 10 mukainen xhtml-macro -moduuli. Moduuli määrittää makron nimen ja kuvauksen, jotka näkyvät Confluence-järjestelmän makrohallinnointinäkymässä, kun käyttäjä luo uuden sivun tai muokkaa olemassa olevaa sivua. Tärkein moduulin attribuutti on class, joka makron määrittää Java-luokkatiedoston. Makron Java-luokkatiedosto ajetaan, kun makro lisätään Confluence sivulle.

```
<xhtml-macro name="meeting Rooms"
             class="com.ambientia.meetingrooms.plugin.MeetingRoomsMacro"
             key='meetingrooms-macro'>
    <description>Shows meeting rooms reservation status.</description>
    <parameters/>
</xhtml-macro>
```

Ohjelmakoodi 10. Xhtml-macro -moduulin määrittäminen.

Makron Java-luokkatiedosto näyttää ohjelmakoodin 11 mukaiselta. Se sisältää tarvittavat riippuvuudet Atlassian Macro luokan käyttöön ja määrittämiseen. Luokan on perittävä Atlassian Macro luokka ja sisältää ohjelmakoodissa 11 kuvatut oletusmetodit. Tärkein metodeista on execute. Tämä metodi sisältää toiminnot jotka halutaan suorittaa, kun makro lisätään sivulle. Metodin tulee palauttaa merkkijonomuuttuja, jotta saadaan makron kautta sivulle html-tietomuodossa olevaa dataa, se kirjoitetaan tässä merkkijono muotoon ja palautetaan metodissa return-lausekkeella. Makron HTML-tiedon rakentamiseen käytetään .vm-tiedostoa, joka sisältää käyttöliittymän HTML-elementit. Tiedosto sijaitsee pluginin resursseissa templates-kansiossa ja tässä makron Java-luokkatiedostossa on muuttuja, joka sisältää .vm-tiedoston polun. Muuttujaa käyttämällä voin palauttaa .vm-tiedoston suoraan metodin suorituksen aikana.

```
package com.ambientia.meetingrooms.plugin;

import com.atlassian.confluence.content.render.xhtml.ConversionContext;
import com.atlassian.confluence.macro.Macro;
import com.atlassian.confluence.macro.MacroExecutionException;
import com.atlassian.confluence.renderer.radeox.macros.MacroUtils;
import com.atlassian.confluence.util.velocity.VelocityUtils;

import java.util.Map;

public class MeetingRoomsMacro implements Macro {

    private static final String MACRO_BODY_TEMPLATE = "templates/meetingrooms.vm";

    public String execute(Map<String, String> map, String s, ConversionContext conversionContext)
     throws MacroExecutionException {

        Map<String, Object> context = MacroUtils.defaultVelocityContext();

        return VelocityUtils.getRenderedTemplate(MACRO_BODY_TEMPLATE, context);
    }

    public BodyType getBodyType() { return BodyType.NONE; }

    public OutputType getOutputType() { return OutputType.BLOCK; }
}
```

Ohjelmakoodi 11. MeetinRoomsMacro.java tiedosto.

Velocity kaavainjärjestelmän .vm-tiedosto sisältää käyttöliittymän HTML-tiedot ohjelmakoodin 12 mukaisesti. HTML koostuu yksinkertaisista div-elementeistä, jotka ovat muokattu pluginin suunnittelussa määritetyiksi värineliöiksi. Jokainen elementti sisältää ”id”-attribuutin, jonka avulla voidaan tehdä muutoksia elementteihin JavaScriptin avulla. Ohjelmakoodin 12 .vm-tiedoston ensimmäisessä lausekkeessa määritän, että tämä .vm vaatii resurssipaketin, joka on valmiiksi määritetty Confluence plugin projektin luomisen aikana. Tämä resurssipaketti sisältää meetingrooms.css-tyylitiedoston, jonka avulla makron HTML-elementtien ulkonäkö on määritetty. Resurssipaketti sisältää myös meetingrooms.js-tiedoston, jota käytetään makron dynaamisen päivityksen toteuttamiseen.

```
#requireResource("com.ambientia.meetingrooms.plugin.meetingrooms:meetingrooms-resources")

<div class="rooms-container">
  <div class="room" id="YELLOWSTONE"></div>
  <div class="room" id="BLUESCREEN"></div>
  <div class="room" id="GREENHOUSE"></div>
  <div class="room" id="REDZONE"></div>
</div>
```

Ohjelmakoodi 12. meeting-rooms.vm kaavaintiedosto.

JavaScript-tiedosto, joka on määritetty meetingrooms-resources -pakettiin, sisältää ohjelmakoodin 13 mukaista koodia ja toiminnallisuutta. Koodi on pelkistettyä ja toiminnallisuudet tarjoavat vain ChangeColor-metodin, jonka avulla voidaan testata makron dynaamista päivittämistä. Metodi vaihtaa koodissa määritetyn ”YELLOWSTONE”-elementin luokkalistaa,

jonka perusteella värineliön väri vaihtuu makrossa. Toiminto saadaan toistumaan viiden sekunnin välein käyttämällä JavaScriptin setInterval-metodia, joka ottaa vastaan parametreinä suoritettavan metodin ja toiston aikavälin millisekunteina.

```
document.addEventListener('DOMContentLoaded', function(){

  var reservedClass = "reserved";

  setInterval(ChangeColor, 5000);

  function ChangeColor () {

    var roomElement = document.getElementById('YELLOWSTONE');

    if (roomElement.classList.contains(reservedClass))
      roomElement.classList.remove(reservedClass);
    else
      roomElement.classList.add(reservedClass);
  }

}, false);
```

Ohjelmakoodi 13. meetingrooms.js dynaamisen päivityksen testausversio.

Makron dynaaminen päivitys onnistuu erittäin hyvin JavaScriptin avulla.

8.2 Servletin luominen ja yhteys makroon

Seuraavaksi lähestytään pluginin servlet-moduulin kehittämistä. Servlet hoitaa pluginin toiminnallisuudelle määritetyt vaatimukset, jotka tulee suorittaa palvelimen puolella. Servletin tarkoitus on suorittaa HTTP-pyynnöt asiakkaan Zimbra kalenteriin käyttämällä Zimbra-järjestelmän REST-ohjelmointirajapintaa.

Jotta voidaan luoda Confluence plugiin servlet, täytyy pluginin atlassian-plugin.xml tiedostoon lisätä ohjelmakoodin 14 mukainen servlet-moduuli. Moduuli määrittää komponentin nimen ja class-attribuutissa servletin Java-luokkatiedoston, joka on Java-pohjainen servletin toteutus. Moduuli sisältää myös url-pattern -elementin, joka on välttämätön servlet-moduulille. Elementti määrittää servletin sijainnin Confluence-järjestelmässä. Koska elementin arvona on ”meetingroomsservlet”, on servletin sijainti Confluence-järjestelmässä ”*/confluence/plugins/servlet/meetingrooms-servlet”. Testausympäristö pyörii paikallisella palvelmille ja portissa 1900 eli servletin osoite on ”http://localhost:1990/confluence/plugins/servlet/meetingroomsservlet”.

```
<servlet name="meeting Rooms Servlet"
         key="meetingRoomsServlet"
         class="com.ambientia.meetingrooms.plugin.MeetingRoomsServlet">
    <description>Meeting Rooms Servlet returns room reservation information.</description>
    <url-pattern>/meetingroomsservlet</url-pattern>
</servlet>
```

Ohjelmakoodi 14. servlet-moduulin määrittäminen.

Servletin Java-luokkatiedosto sisältää ohjelmakoodin 15 mukaiset riippuvuudet ja määritykset, joiden avulla voidaan toteuttaa Java-pohjainen servlet. Nämä riippuvuudet ulkoisista paketeista tulee merkitä plugin projektin pom.xml tiedostoon, jotta plugin voi käyttää niitä. MeetingRooms-Servlet-tiedosto sisältää servletille ominaisen doGet-metodin. DoGet-metodi ajetaan, kun jokin lähde tekee HTTP-pyynnön tähän servletiin. Servletin toimivuutta testataan kokeilemalla merkkijonon palauttamista, jota HTTP-pyynnön tehnyt lähde voisi käsitellä. Tässä versiossa, servlet palauttaa doGet-metodin suorituksessa satunnaisesti arvotun neuvotteluhuoneen nimen meetingRooms-taulukosta kutsun lähettäneelle lähteelle.

```
package com.ambientia.meetingrooms.plugin;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.util.Random;

public class MeetingRoomsServlet extends HttpServlet {

  String[] meetingRooms = {"YELLOWSTONE",
                           "BLUESCREEN",
                           "GREENHOUSE",
                           "REDZONE"};

  Random generator = new Random();

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                       throws ServletException, IOException {

    resp.setContentType("text/plain");
    resp.setCharacterEncoding("UTF-8");
    resp.getWriter().write(meetingRooms[generator.nextInt(3)]);
  }
}
```

Ohjelmakoodi 15. MeetingRoomsServlet.java tiedoston kokeiluversio.

JavaScript-tiedostoa meetingrooms.js on muokattu ohjelmakoodin 16 mukaisesti. Tiedosto tarvitsee uuden muuttujan, joka sisältää Confluence-järjestelmän servletin sijainnin. Muuttujan avulla voidaan tehdä HTTP-

pyyntö, käyttämällä luotua requestMeetingRoomsServlet-metodia. Metodi lähettää GET-pyynnön servletiin servletpath-muuttujan avulla. Kun HTTP-pyyntö on onnistunut, voidaan käsitellä pyynnöstä saatua vastausta eli arvottua huoneen nimeä. Vastauksen avulla ohjelma muuttaa makron värineliön väriä käyttämällä ChangeColor-metodia.

```
var reservedClass = "reserved";
var servletpath = "http://localhost:1990/confluence/plugins/servlet/meetingroomsservlet";

setInterval(requestMeetingRoomsServlet, 30000);

function requestMeetingRoomsServlet () {

  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {

      ChangeColor(this.responseText);
    }
  };
  xhttp.open("GET", servletpath, true);
  xhttp.send();
}

function ChangeColor (roomname) {
  var roomElement = document.getElementById(roomname);

  if (roomElement.classList.contains(reservedClass))
    roomElement.classList.remove(reservedClass);
  else
    roomElement.classList.add(reservedClass);
}
```

Ohjelmakoodi 16. Päivitetty meetingrooms.js tiedosto http-pyyntöjen tekemiseen ja vastauksen käsittelyyn.

8.3 Pyyntöjen tekeminen Zimbra REST-ohjelmointirajapintaan

Seuraavaksi aloitetaan pluginin toiminnallisuudelle erittäin merkittävän ominaisuuden kehittäminen ja soveltaminen servletiin. Tavoitteena on toteuttaa ohjelmakoodi, joka tekee GET-tyyppisen HTTP-pyynnön asiakkaan Zimbra-järjestelmän REST-ohjelmointirajapintaan.

Ennen kuin lähdetään toteuttamaan tämän metodin toteutusta servletiin, luodaan ohjelmakoodin 17 mukaisen Java-luokkatiedoston nimeltä MeetinRoom.java. Luokka esittää neuvotteluhuonetta ja sisältää neuvotteluhuoneeseen liittyviä tietoja, joita tarvitaan HTTP-pyyntöjen tekemiseen ja makron värineliöiden päivittämiseen. MeetingRoom-luokka on yksinkertainen ja sisältää kolme muuttujaa ja niiden käyttöön tarvittavat get- ja set-metodit. MeetingRoom-luokan name-muuttujan avulla voidaan makron puolella poimia oikean värineliön käsiteltäväksi ja reserved-muuttuja

sisältää boolean-arvon, jonka perusteella värineliön tila on joko varattu tai vapaa.

```
package com.ambientia.meetingrooms.plugin;

public class MeetingRoom {
  private String name;
  private String account;
  private boolean reserved = false;

  public MeetingRoom(String n, String a) {
    this.name = n;
    this.account = a;
  }

  public String getName() { return this.name; }
  public void setName(String n) { this.name = n; }

  public String getAccount() { return this.account; }
  public void setAccount(String a) { this.account = a; }

  public boolean getReserved() { return this.reserved; }
  public void setReserved(boolean ir) { this.reserved = ir; }
}
```

Ohjelmakoodi 17. MeetingRoom.java Java-luokkatiedosto.

Servletiin tarvitaan uusia muuttujia, jotta voidaan toteuttaa Zimbra-järjestelmän REST-ohjelmointirajapintaa HTTP-pyyntöjä tekevä metodi. Servletiin on lisätty ohjelmakoodin 18 mukaisia uusia muuttujia. Tarvitaan muuttuja, joka sisältää polun jonne HTTP-pyyntöjä halutaan tehdä. Muuttujat username ja password ovat välttämättömiä, jotta HTTP-pyyntöjen teko on mahdollista. Zimbra REST-ohjelmointirajapinnan Get FreeBusy - metodi vaatii autentikoinnin. Muuttuja meetingRooms on MeetingRoom-olioita sisältävä taulukko, johon on alustettu asiakkaan toimiston neuvotteluhuoneentietoja kuten nimi ja käyttäjätunnus. Taulukkoa käytetään HTTP-pyyntöjen tekemisessä ja myös kun tieto palautetaan tähän servletiin pyyntöä tekevään lähteeseen.

```
public class MeetingRoomsServlet extends HttpServlet {

  private String path = "https://zimbra-testipalvelin.fi/service/home/kayttajatunnus/";
  private String username = "kayttajatunnus";
  private String password = "salasana";

  MeetingRoom[] meetingRooms = {new MeetingRoom("YELLOWSTONE", "yellowstone@esimerkki.fi"),
                                new MeetingRoom("BLUESCREEN", "bluescreen@esimerkki.fi"),
                                new MeetingRoom("GREENHOUSE", "greenhouse@esimerkki.fi"),
                                new MeetingRoom("REDZONE", "redzone@esimerkki.fi")};
```

Ohjelmakoodi 18, MeetingRoomsServlet.java luokan uusia muuttujia.

Seuraavaksi toteutetaan ohjelmakoodin 19 mukainen metodi servletiin nimeltä httpClientGetIFBFiles. Tämä metodi tekee HTTP-pyynnöt Zimbra REST-ohjelmointirajapintaan, käsittelee palautetun datan ja tekee

tarvittavia muutoksia servletin meetingRooms-neuvotteluhuonetaulukkoon. Metodi rakentaa HTTP-pyyntöön tarvittavan URI-osoitteen käyttämällä URIBuilder-luokkaa. Jotta Get FreeBusy -metodin pyyntö on halutun kaltainen, tarvitsee URI-osoitteeseen lisätä parametrejä ja autentikointi ”HEADER”, jotta pyynnön tekeminen on sallittu. URIBuilder-luokan setParameter-metodin avulla voidaan URI-osoitteeseen lisätä parametrejä. Halutaan tehdä pyyntö, joka palauttaa ifb-tiedoston. Lisätään pyyntöön fmt-parametri, jonka arvoksi määritetään ifb. Parametreiksi lisätään myös aloitusaika ja lopetusaika, jotka määräävät miltä aikaväliltä Get FreeBusy -metodi hakee tiedot. Aloitusajaksi määritetään nykyinen aika ja lopetusajaksi minuuttia myöhempi aika. Jotta aikaparametri toimivat pyynnössä, täytyy ne muuttaa millisekunneiksi.

```
private void httpClientGetIFBFiles() throws Exception {

  Date curDate = new Date();
  String curTimeMillis = String.valueOf(curDate.getTime());

  for (MeetingRoom room: meetingRooms) {
    CloseableHttpClient httpclient = HttpClients.createDefault();

    URIBuilder builder = new URIBuilder();
    builder.setPath(path + room.getName())
    .setParameter("fmt", "ifb")
    .setParameter("start", curTimeMillis)
    .setParameter("auth", "ba,nsc,qp");
    URI uri = builder.build();

    HttpGet httpGet = new HttpGet(uri);

    httpGet.addHeader("Authorization", this.getB64Auth());

    CloseableHttpResponse response = httpclient.execute(httpGet);

    String IFBAsString = "";

    try {
      HttpEntity entity = response.getEntity();
      IFBAsString = EntityUtils.toString(entity);
    } catch (Exception e) {
      System.out.print(e.toString());
    } finally {
      response.close();
      httpclient.close();
    }

    this.updateMeetingRoomReservedStatus(room, IFBAsString);
  }
 }
```

Ohjelmakoodi 19. httpClientGetIFBFiles-metodi.

Zimbra REST-ohjelmointirajapintaan tehdystä HTTP-pyynnöstä tallennetaan data IFBAsString-muuttujaan merkkijonomuotoon, joten sitä voidaan helposti käsitellä. Pyynnön palauttama data on ohjelmakoodin 20 mukainen .ifb-tiedosto. Tiedosto sisältää tietoja HTTP-pyynnön ajankohdasta ja kaikista tietysti tiedon siitä onko kalenterissa tapahtumia tällä hetkellä.

```
BEGIN:VCALENDAR
PRODID:Zimbra-Calendar-Provider
VERSION:2.0
METHOD:PUBLISH
BEGIN:VFREEBUSY
ORGANIZER:mailto:cf-meetingroom-plugin@zimbra-sandbox.ambientia.fi
DTSTAMP:20171219T115923Z
DTSTART:20171219T115833Z
DTEND:20171219T120833Z
URL:https://"https://zimbra-testipalvelin.fi/service/home/kayttajatunnus/
          REDZONE?fmt=ifb&start=1513684713228&end=1513685313228&auth=ba,nsc,qp
FREEBUSY;FBTYPE=BUSY:20171219T115833Z/20171219T120833Z
END:VFREEBUSY
END:VCALENDAR
```

Ohjelmakoodi 20. ifb-tiedosto esimerkki.

Ohjelmakoodista 20 voidaan nähdä ”FREEBUSY;” merkkijonolla alkava rivi ”URL:”-tiedon sisältävän rivin alapuolella. Tämä rivi kertoo, onko kalenterissa merkintöjä http-pyynnön yhteydessä annetulla aikavälillä. Jos rivi sisältää merkkijonon ”FREEBUSY;FBTYPE=BUSY”, on kalenterissa merkintä ja neuvotteluhuone on varattuna. Jos riviä ei löydy, merkintää ei ole kalenterissa ja neuvotteluhuone on tällöin vapaana. Tämän kaltainen tarkastus tehdään servletissä ohjelmakoodin 21 kaltaisella metodilla. Metodi päivittää huoneen boolean tyyppisen reserved-muuttujan arvoa, false jos neuvotteluhuone on vapaana ja true jos neuvotteluhuone on varattuna.

```
private void updateMeetingRoomReservedStatus(MeetingRoom room, String ifb) {
  String ifbstatus = "FREEBUSY;FBTYPE=BUSY";

  if (ifb.contains(ifbstatus)) room.setReserved(true);
  else room.setReserved(false);
}
```

Ohjelmakoodi 21. updateMeetingRoomReservedStatus-metodi.

Kun ohjelmakoodin 19 httpClientGetIFBFiles-metodi on hakenut kaikkien neuvotteluhuoneiden .ifb-tiedostot, käsitellyt ne ja päivittänyt neuvotteluhuoneiden varaustilannetta säilyttävän muuttujan arvon, luodaan meetingRooms-taulukosta kopio ja käännetään se JSON-tietomuotoon. Ohjelmakoodin 22 mukaisesti merkkijono meetingRoomsJSON-muuttujaan kopioidaan meetingRooms-taulukon tiedot Gson-luokan avulla. Gson-luokan toJson-metodilla voidaan melkein mikä tahansa olio muuttaa merkkijonoksi, joka sisältää olion sisällön JSON-tietomuodossa. Tämän jälkeen palautetaan meetingRoomsJSON-muuttuja HTTP-pyyntöä tekevään lähteeseen merkkijonona.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {

  try {
    this.httpClientGetIFBFiles();
  } catch (Exception e) {
    System.out.print(e.toString());
  }

  String meetingRoomsJSON = new Gson().toJson(meetingRooms);

  resp.setContentType("text/plain");
  resp.setCharacterEncoding("UTF-8");
  resp.getWriter().write(meetingRoomsJSON);
}
```

Ohjelmakoodi 22. MeetingRoomsServlet.java tiedoston meetingRooms-taulukon kääntämien JSON-muotoon.

Ohjelmakoodin 22 doGet-metodi suoritetaan, kun jostain tehdään GET-tyyppinen HTTP-pyyntö tämän servletin osoitteeseen, palauttaa ohjelmakoodin 23 mukaisen JSON-tietomuotoisen merkkijonon. Tätä merkkijonoa käyttämällä pluginin JavaScript-tiedosto osaa päivittää käyttöliittymän oikean näköiseksi.

```
[{"name":"YELLOWSTONE","account":"yellowstone@esimerkki.fi","reserved":false},
 {"name":"BLUESCREEN","account":"bluescreen@esimerkki.fi","reserved":false},
 {"name":"GREENHOUSE","account":"greenhouse@esimerkki.fi","reserved":false},
 {"name":"REDZONE","account":"redzone@esimerkki.fi","reserved":false}]
```

Ohjelmakoodi 23. JSON-muotoon käännetty meetingRooms-taulukon sisältö.

Pluginin JavaScript-tiedostossa oleva HTTP-pyyntö käsittelyyn tarkoitettu onreadystatechange-metodi on päivitetty ohjelmakoodin 24 mukaiseksi. Kun HTTP-pyyntö on onnistunut, saadaan vastauksena merkkijono, joka voidaan kääntää JSON-tietomuotoiseksi JSON-luokan parse-metodilla. Tällä tavoin palautetun tiedon käsittely on helppoa. Jokainen rooms-muuttujassa sijaitsevan MeetingRoom-olio käsitellään toistolausekkeessa ohjelmakoodin 25 mukaisella updateRoomInClient-metodilla.

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {

    var rooms = JSON.parse(this.response);

    for (var i = 0; i < rooms.length; i++) updateRoomInClient(rooms[i]);
  }
```

Ohjelmakoodi 24. Päivitetty onreadystatechange-metodi.

Ohjelmakoodin 25 updateRoomInClient-metodi ottaa vastaan yhden parametrin, vastauksen saadun JSON-tiedostossa olevan MeetingRoom-olion. Tämän olion name-muuttujan avulla haetaan makrosta vastaava värineliö. Värineliön id-attribuutti on sama kuin neuvotteluhuoneen nimi. Kun värineliö elementti on haettu, tarkistetaan ehtolausekkeella mikä on olion reserved-attribuutin arvo. Jos reserved on arvoltaan ”true” eli huone on varattu, lisätään elementin luokkalistaan reservedClass-tyyliluokka, jos se on tarpeellista. Jos taas huone on vapaana, reservedClass-tyyliluokka poistetaan elementin luokkalistasta.

```
function updateRoomInClient (room) {

  var roomElement = document.getElementById(room.name);

  if (room.reserved) {
    if (!roomElement.classList.contains(reservedClass))
      roomElement.classList.add(reservedClass);
  } else {
    if (roomElement.classList.contains(reservedClass))
      roomElement.classList.remove(reservedClass);
  }
}
```

Ohjelmakoodi 25. Värineliöiden tilan muuttumisen updateRoomInClient-metodi.

8.4 Varauksen lisätiedot makroon

Pluginin kehityksen tässä vaiheessa, asiakkaan kanssa keskusteltaessa tuli esille lisäominaisuus pluginin makroon. Ominaisuus olisi seuraavanlainen: kun käyttäjä siirtää hiirensä makrossa olevan värineliön päälle, näyttää ”tooltip”-viesti värineliön edustavan neuvotteluhuoneen varaustilanteen tietoja. Jos neuvotteluhuone on vapaana, ”tooltip”-viestissä lukee "Available". Neuvotteluhuoneen ollessa varattuna, 'tooltip'-viestissä lukisi varauksen tekijän etu- ja sukunimi ja varauksen ajankohta. Viesti olisi muotoa "Reserved: Etunimi Sukunimi 12:00 - 13:00". Ongelmana on, että Zimbra REST-ohjelmointirajapinnan Get FreeBusy -metodi ei ilmoita kalenteritapahtuman aikaväliä, joten joudutaan käyttämään eri ohjelmointirajapinnan metodia, joka palauttaa .ics-muodossa olevan datan.

Zimbra REST-ohjelmointirajapinnan Get Calender -metodi palauttaa .ics-tietomuotoa olevan tiedoston. Tämä tiedosto sisältää HTTP-pyynnön parametrein määritetyn kalenterin tietoja, kuten aikaväli parametrien ajankohdalle sijoittuvat kalenteritapahtumat. Zimbra REST-ohjelmointirajapinnan Get Calender -metodia voidaan käyttää samalla ohjelmakoodin 19 mukaisella metodilla, mutta HTTP-pyynnön URI-osoitteessa määritetyn fmt eli format, arvo on oltava .ics eikä .ifb. Muokataan ohjelmakoodin 19 metodin äskeisen kuvauksen mukaiseksi ja uudelleen nimetään metodi ”httpClientGet.ICSFiles”. Servletin ei tarvitse enään hakea ifb-tiedostoja, koska saadaan kalenterin varaustilanne selville myös .ics-tiedostosta. Jos

kalenterissa ei ole kalenteritapahtumia parametreillä määritetyllä aikavälillä, palautettava tiedosto sisältää ohjelmakoodin 26 mukaisen lyhyen tiedoston, joka sisältää perustietoja pyynnössä määritetystä kalenterista.

```
BEGIN:VCALENDAR
X-WR-CALNAME:REDZONE
X-WR-CALID:88620951-da9b-470a-a27d-598faaa24944:282
PRODID:Zimbra-Calendar-Provider
VERSION:2.0
METHOD:PUBLISH
END:VCALENDAR
```

Ohjelmakoodi 26. .ics-tiedosto, jossa ei ole kalenteritapahtumia.

Jos kalenterissa on kalenteritapahtumia, löytyy .ics-tiedostosta ohjelmakoodin 26 mukainen osio, joka alkaa merkkijonorivillä ”BEGIN:VEVENT” ja päättyy merkkijonorivillä ”END:VEVENT”. Näiden rivien väliltä löytyy kalenteritapahtumaan liittyviä tietoja, joista tässä kappaleessa toteuttavaan ominaisuuteen tärkeimpiä ovat kalenteritapahtuman varaajan tiedot, kalenteritapahtuman ajankohtaan liittyvät tiedot ja mikä on kalenteritapahtuman varaustilanne eli ”FREE/BUSY”-tieto.

Ohjelmakoodin 27 ”ORGANIZER”- merkkijonolla alkava rivi sisältää kalenteritapahtuman tehneen käyttäjän sähköpostiosoite. Tämä sähköposti on asiakkaan organisaatiossa muotoa ”etunimi.sukunimi@asiakas.fi”. Tästä sähköpostin sisältävästä rivistä voin siis hakea varaan etu- ja sukunimen varauksen lisätietoihin. Seuraavilta riveiltä löytyy kalenteritapahtuman ajankohtatiedot, ”DTSTART” sisältää aloitus ajankohdan ja ”DTEND” sisältää loppumisajankohdan. Saan rakennettua varauksen ajankohdan näiltä riveiltä varauksen lisätietoihin. Vielä on hyvä toteuttaa kalenteritapahtuman varaustilanteen tarkistus ”X-MICROSOFT-CDO-INTENTEDSTATUS”-merkkijonolla alkavalta riviltä, koska vaikka kalenterissa onkin kalenteritapahtuma, voi varauksen ”FREE/BUSY”-tieto olla määritettynä ”FREE”-tietona eli vapaana.

```
BEGIN:VEVENT
UID:99c5360f-5665-47e6-b3c0-39315e4bb193
SUMMARY:Varaus
ORGANIZER:mailto:cf-meetingroom-plugin@zimbra-sandbox.ambientia.fi
DTSTART;TZID="Europe/Athens":20171222T070000
DTEND;TZID="Europe/Athens":20171222T080000
STATUS:CONFIRMED
CLASS:PUBLIC
X-MICROSOFT-CDO-INTENDEDSTATUS:BUSY
TRANSP:OPAQUE
LAST-MODIFIED:20171222T050635Z
DTSTAMP:20171222T050635Z
SEQUENCE:0
BEGIN:VALARM
ACTION:DISPLAY
TRIGGER;RELATED=START:-PT5M
DESCRIPTION:Reminder
END:VALARM
END:VEVENT
```

Ohjelmakoodi 27. .ics-tiedosto, jossa on kalenteritapahtuma ja kalenteritapahtumaan liittyviä tieoja.

Varauksen lisätietojen tallentamisen mahdollistamiseen, lisätään MeetingRoom.java-tiedostoon ohjelmakoodin 28 mukaiset uudet reservationInfo-muuttuja ja muuttujalle vastaavat get- ja set-metodit muuttujan tietojen muuttamiseen ja noutamiseen. Muuttujan arvona on oletuksena merkkijono "Available". Muuttujan arvoa päivitetään luomillani apumetodeilla, kun haetun .ics-tiedoston sisältöä käsitellään. Apumetodit käsittelevät sähköpostin merkkijonoksi, joka sisältää varauksen tehneen käyttäjän nimen halutussa muodossa ja aikatiedot haluttuun muotoon.

```
package com.ambientia.meetingrooms.plugin;

public class MeetingRoom {
  private String name;
  private String account;
  private boolean reserved = false;
  private String reservationInfo = "Available"; //Reserved: Etunimi Sukunimi 11:30 - 12:45
```

Ohjelmakoodi 28. Uusi reservationInfo-muuttuja MeetingRoom-luokassa.

Pluginin JavaScript-tiedoston makron käyttöliittymän päivittämiseen käytettävää metodia muokataan ohjelmakoodin 29 mukaisesti. Metodi lisää HTML-elementin title-attribuuttiin huoneen nimen ja reservationInfo-muuttujan sisällön.

```
function updateRoomInClient (room) {

  var roomElement = document.getElementById(room.name);

  if (room.reserved) {

  roomElement.title = room.name + "\n" + room.reservationInfo;
}
```

Ohjelmakoodi 29. updateRoomInClient-metodi lisää HTML-elementin title-attribuuttiin varauksen lisätiedot.

Makron käyttöliittymässä tulee esiin tooltip-viestinä huoneen nimi ja varauksen lisätiedot. Tooltip-viesti tulee näkyviin kuvan 18 mukaisesti, kun käyttäjä siirtää hiiren värineliön päälle.

Kuva 18. Varattu värineliö näyttää tooltip-viestinä varauksen lisätiedot.

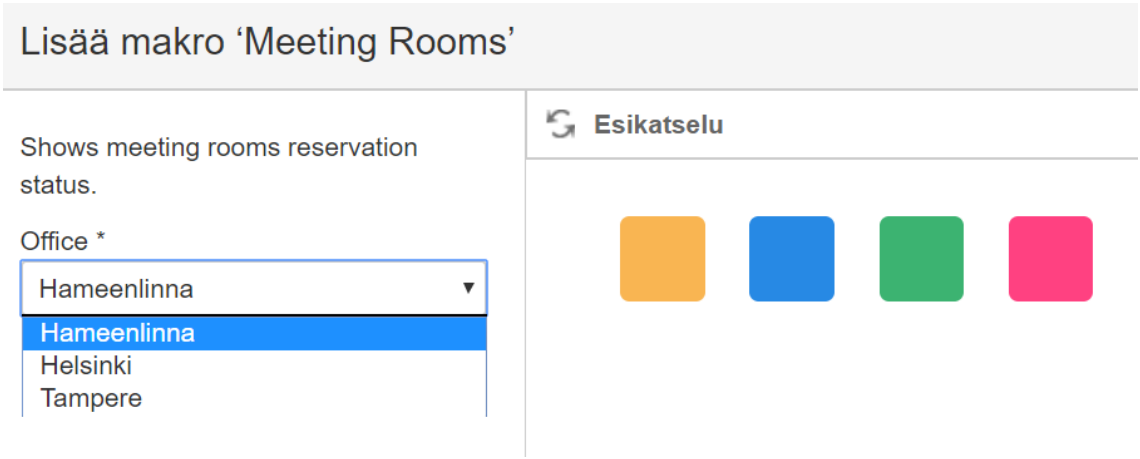
8.5 Toimistot

Ensimmäisessä asiakastapaamisessa keskusteltiin lyhyesti mahdollisuudesta toteuttaa plugin niin, että se olisi käyttöönotettavissa asiakkaan yrityksen muillakin toimistoilla. Tässä vaiheessa tämän Confluence plugin kehitystä, seuraava lisätoiminnallisuuden toteuttaminen olisi muiden toimistojen valinnan lisääminen. Confluence makroon voidaan lisätä parametrejä ohjelmakoodin 30 mukaisella tavalla. Käyttäjän voi makron sivulle lisäämisen yhteydessä siis antaa tai valita arvoja, joiden avulla makron Java-luokkatiedoston toiminnallisuutta voidaan laajentaa. Ohjelmakoodissa 30 on lisätty atlassian-plugin.xml tiedoston makromoduulin määrityksiin Office-parametri elementti. Tämä parametri on tyypiltään enum eli lista, joka koostuu vakioarvoista. Parametrin määrityksissä on annettu required-attribuutille arvo ”true” eli parametri ei voi olla tyhjä. Oletusarvo enum-tyyppisillä parametreillä on listan ensimmäinen arvo eli tässä tapauksessa ”Hameenlinna”.

```
<xhtml-macro name="meeting Rooms"
             class="com.ambientia.meetingrooms.plugin.MeetingRoomsMacro"
             key='meetingrooms-macro'>
    <description>Shows meeting rooms reservation status.</description>
    <parameters>
      <parameter name="Office" type="enum" required="true">
        <value name="Hameenlinna"/>
        <value name="Helsinki"/>
        <value name="Tampere"/>
      </parameter>
    </parameters>
</xhtml-macro>
```

Ohjelmakoodi 30. Parametrejä xhtml-macro -moduulissa.

Parametrin valinta makron lisäyksen yhteydessä näyttää kuvan 19 mukaiselta.



Kuva 19. Office-parametrin arvon valinta makro-hallinnointinäkymässä.

Toimiston valintaa varten luodaan Office.java Java-luokkatiedoston, joka koostuu ohjelmakoodin 31 mukaisesta enum vakio avain-arvo listasta. Office-lista koostuu avaimista, jotka ovat asiakkaan toimistojen sijainteja. Office-listan avainten arvoina on taulukko MeetingRoom-olioita eli neuvotteluhuoneen esiintymiä. Office-listan avainten arvojen noutamiseen on luotu julkinen getValue-metodi, joka palauttaa MeetingRoom-oliosta koostuvan taulukon.

```
package com.ambientia.meetingrooms.plugin;

import com.ambientia.meetingrooms.plugin.MeetingRoom;

public enum Office {

    HAMEENLINNA(new MeetingRoom[]
                {new MeetingRoom("YELLOWSTONE", "yellowstone@esimerkki.fi"),
                 new MeetingRoom("BLUESCREEN", "bluescreen@esimerkki.fi"),
                 new MeetingRoom("GREENHOUSE", "greenhouse@esimerkki.fi"),
                 new MeetingRoom("REDZONE", "redzone@esimerkki.fi")}),

    HELSINKI(new MeetingRoom[]
               {new MeetingRoom("HLESINKI", "helsinki@esimerkki.fi"),

    TAMPERE(new MeetingRoom[]
                {new MeetingRoom("TAMPERE", "tampere@esimerkki.fi"),

    private MeetingRoom[] meetingRooms;

    Office(MeetingRoom[] mrooms) {
        this.meetingRooms = mrooms;
    }

    public MeetingRoom[] getValue() {
        return meetingRooms;
    }
}
```

Ohjelmakoodi 31. Office.java enum tiedosto.

Ohjelmakoodin 31 Office-enum-listaa käyttämällä voidaan helposti käyttäjän makron lisäämisen yhteydessä antaman parametrin avulla noutaa haluttu kokoelma neuvotteluhuoneita. Pluginin makron MeetingRoomsMacro.java Java-luokkatiedoston execute-metodissa alustetaan map-avainarvotaulukko, joka sisältää makrossa määritetyt parametrit. Tämän avainarvotaulukon avulla voidaan noutaa käyttäjän valitsema Office-parametrin arvo ohjelmakoodin 32 mukaisella tavalla. Parametrin noutamisen jälkeen haetaan Office-enum-listasta oikea neuvotteluhuonetaulukko ja sijoitetaan se tiedostossa alustettuun meetinRooms-taulukkoon.

Tässä vaiheessa plugin projektia käytetään .vm-tiedoston ominaisuuksia laajemmin, lisäämällä muuttujia VelocityContext-olioon, jonka kanssa vm-tiedosto käännetään lopulliseen html-muotoon. Lisätään käyttäjän valitsema Office-parametrin arvon jälkeisempää käyttöä varten ja aikaisemmin Office-parametrin avulla määritetyn neuvotteluhuoneista koostuvan taulukon.

```
public String execute(Map<String, String> map, String s, ConversionContext conversionContext)
                    throws MacroExecutionException {

    String officeParam = map.get("Office");

    for (Office ofc : Office.values()){
      String ofcLocation = ofc.name();
      if (ofcLocation.equalsIgnoreCase(officeParam)) {
        meetingRooms = ofc.getValue();
      }
    }

    Map<String, Object> context = MacroUtils.defaultVelocityContext();

    context.put("officeParam", officeParam.toUpperCase());
    context.put("meetingRooms", meetingRooms);

    return VelocityUtils.getRenderedTemplate(MACRO_BODY_TEMPLATE, context);
}
```

Ohjelmakoodi 32. MeetingRoomsMacro.java Office-enum-listan käyttäminen.

Päivitetään meetingrooms.vm-tiedosto ohjelmakoodin 33 mukaiseksi käyttämään tiedoston renderöinnin yhteydessä lisättyä kontekstia. Lisätään tiedostoon html-elementtinä JavaScript-ohjelmointilohkon, jossa on alustettu ja määritetty officeParam-muuttuja. Muuttujan määrityksessä muotoillaan URI-muotoinen parametri ”office=Hameenlinna”, notaatiolla ”$officeParam” haetaan tämän nimisen muuttujan arvoa .vm renderöinnin mukana olevasta kontekstista. Tiedostossa on myös toistolauseke, joka käyttää kontekstiin mukana lisättyä neuvotteluhuoneista koostuvaa taulukkoa. Toistolauseke luo jokaiselle taulukon oliolle värineliön ja määrittää värineliön id-attribuutiksi neuvotteluhuoneen nimen käyttämällä MeetingRoom-olion getName-metodia.

```
#requireResource("com.ambientia.meetingrooms.plugin.meetingrooms:meetingrooms-resources")
<script>
  var officeParam = "office" + "=" + "$officeParam";
</script>

<div class="rooms-container">
  #foreach( $room in $meetingRooms )
      <div class="room" id="$room.getName()" ></div>
  #end
</div>
```

Ohjelmakoodi 33. Päivitetty meetingrooms.vm kaavaintiedosto.

Ohjelmakoodi 33 .vm-tiedostoon lisätyn officeParam-muuttujan arvo käytetään päivitetyssä meetingrooms.js JavaScript-tiedostossa, jossa tehdään HTTP-pyynnöt servletiin ohjelmakoodin 34 mukaisesti. GET-tyyppisen HTTP-pyynnön tekemiseen määritettyyn osoitteeseen lisätään

officeParam-muuttujan arvo eli lisätään pyyntöön tieto valitusta toimistosta, jonka avulla servlet osaa hakea oikeiden neuvotteluhuoneiden varaustilanteet.

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
};
xhttp.open("GET", servletpath + "?" + officeParam, true);
xhttp.send();
```

Ohjelmakoodi 34. meetingrooms.js tiedoston HTTP-pyyntöjen tekeminen officePrama-muuttujan kanssa.

Kun servletissä saadaan GET-tyyppinen HTTP-pyyntö, haetaan pyynnöstä ohjelmakoodin 35 mukaisella tavalla office- parametri, joka sisältää makrossa valitun toimiston paikkakunnan nimen. Office-parametristä saadulla arvolla haetaan Office-enum-listaa käyttämällä oikeat neuvotteluhuoneet sisältävä neuvotteluhuonetaulukko.

```
String officeParam = "";
if (req.getParameter("office") != null){
  officeParam = req.getParameter("office");

  for (Office ofc : Office.values()){
    String ofcLocation = ofc.name();
    if (ofcLocation.equalsIgnoreCase(officeParam)) {
      meetingRooms = ofc.getValue();
    }
  }
```

Ohjelmakoodi 35. officeParam-muuttujan arvon noutaminen servletissä.

8.6 Makron värineliöiden värikulma

Pluginin toiminnallisuuteen liittyvien ominaisuuksien kehittämisen loputtua, keskitytään pluginin makron käyttöliittymän jatkokehitykseen. Kehittämisen aikana tuli ilmi seuraavanlainen ongelma: kun useampi neuvotteluhuone on varattuna, useampi värineliö on täysin harmaansävyisenä ja makrossa olevissa värineliössä ei ole minkäänlaista suoraa visuaalista merkintään siitä, mitkä neuvotteluhuoneet ovat varattuina. Makron käyttöliittymää lähdettiin kehittämään sillä ajatuksella, että värineliöön lisättäisiin jokin visuaalinen merkki, josta käyttäjä osaa tunnistaa mitä huonetta värineliö edustaa silloinkin, kun neuvotteluhuone on varattuna.

Toteutetaan huoneen värin visualisointi niin, että kun neuvotteluhuone on varattuna, lisätään värikulman värineliöön. Värikulma on saman värinen kuin värineliön perusväritys. Värikulma on svg-elementti ja se lisätään ohjelmakoodin 36 mukaisesti meetingrooms.vm kaavaintiedostossa jokaisen värineliö elementin sisälle. Svg-elementti on Scalable Vector Graphic -grafiikka, joka voidaan määrittää HTML-elementtinä web-sivulle.

```
#foreach( $room in $meetingRooms )
    <div class="room" id="$room.getName()" >
      <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 3.9687498 3.9687498"
                height="15" width="15">

        <path class="corner"
           style="fill-opacity:0.99014779;stroke:none;stroke-width:0.26836443px;
                  stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1"
           d="M 0,3.9687498 V 0.92915028 C 0,0.49090868 0.48852298,
              -1.6917597e-8 0.93158262,-1.6917597e-8 H 3.9687498 Z"
           id="corner-path"/>
      </svg>
    </div>
#end
```

Ohjelmakoodi 36. Värikulma svg-elementin lisääminen värineliöön meetingroom.vm-tiedostossa.

Svg-elementin ominaisuuksia, kuten muidenkin HTML-elementtien ominaisuuksia, voidaan käsitellä ja muokata css-tyylitiedostoissa. Svg-elementin sisäiselle path-elementille class-attribuutiksi on annettu ohjelmakoodin 36 mukaisesti arvo ”corner”. Tällä arvolla voidaan määrittää tyylisääntöjä tälle värikulmalle css-tyylitiedostossa ohjelmakoodin 37 mukaisesti. Ohjelmakoodissa 37 on määrätty värineliön eli neuvotteluhuoneen, ”REDZONE” taustavärin arvoksi ”#ff4181” ja samalla tämän värineliö elementin lapsena olevan ”corner”-luokan värikulman täyttöväriksi sama arvo.

```
#REDZONE, #REDZONE .corner {
  background-color: #ff4181;
  fill: #ff4181;
}
```

Ohjelmakoodi 37. Värikulman värin määritys css-tyylitiedostossa.

Valmis lopputulos on kuvan 20 mukainen.

Kuva 20. Toteutettu värikulma makron värinelilöissä.

9 YHTEENVETO

Confluence pluginin suunnittelu ja toteutus onnistui erinomaisesti tämän työn teoriataustan avulla. Confluence plugin on rakenteeltaan yksinkertainen ja pluginin kehitystyö on helposti lähestyttävä kohtuullisella Java-ohjelmointiosaamisella. Atlassian SDK -kehittäjäpaketin työkalujen käyttäminen pluginin kehityksen ohella tuotti aluksi vaikeuksia Confluence Demonstration Space -demonstrointitilan kanssa, lähinnä sulkemisen osalta. Sulkeminen väärillä tavoilla estää demonstrointitilan uudelleen suorittamisen, koska aiemmasta suorittamisesta on jäänyt Java JVM -taustaohjelmia pyörimään. Muuten Atlassian SDK -kehittäjäpaketin työkalut toimivat moitteettomasti pluginin kehitystyössä.

Asiakkaan pyynnöstä toteutettiin kappale Atlassian SDK -kehittäjäpaketin työkalujen käyttämisestä ja myös lyhyt ”Hello World” plugin-esimerkki. Esimerkissä käydään läpi tarvittavien työkalujen käyttöä, plugin-projektin luominen sekä makropluginin kehittäminen ja käyttöönotto.

Asiakkaalle kehitetty Confluence plugin on tämän työn aikana toteutettu suunnittelussa määritetyiltä toiminnallisuuksiltaan täysin. Pluginin suunnittelu ja toteutus onnistuivat hyvin ja työlle määrätyn aikataulun puitteissa. Pluginia kehitettäessä oltiin yhteydessä asiakkaan vastuuhenkilön kanssa suunnittelun ja toteutuksen aikana. Vastuuhenkilön kanssa työskentelyn ansiosta Zimbra REST-ohjelmointirajapinnan pyyntöjen ja tiedonkäsittelyn oppiminen ja hyödyntäminen pluginin ominaisuuksissa oli helpompaa ja nopeampaa. Vastuuhenkilön kanssa keskusteltiin myös mahdollisista lisäominaisuuksista pluginiin ja onnistuttiin toteuttamaankin nämä lisäominaisuudet projektin ajan puitteissa. Neuvotteluhuoneiden varaustilannetta tarkkailevaa pluginia ei ole vielä käyttöönotettu tai asennettu asiakkaan Confluence-järjestelmään.

Työn ansiosta Java-ohjelmointiosaamiseni on kehittynyt valtavasti ja erityisesti hallitsen paremmin REST-ohjelmointirajapintojen käyttämisen Java-ohjelmoinnissa. Confluence-järjestelmän pluginien kehittäminen oli minulle ennen tätä työtä täysin tuntematonta, joten jouduin lukemaan paljon Confluence-järjestelmän ohjelmakoodidokumentteja. Ohjelmakoodidokumenttien lukemisessa olen kehittynyt paljon tämän työn aikana.

LÄHTEET

Apache Software Foundation. (n.d.a.). Velocity Project.
Haettu 26.12.2017 osoitteesta
https://velocity.apache.org/

Apache Software Foundation. (n.d.b.). Velocity Project Overview.
Haettu 26.12.2017 osoitteesta
https://velocity.apache.org/engine/2.0/overview.html

Atlassian. (n.d.a.). Confluence – Team Collaboration Software.
Haettu 4.11.2017 osoitteesta
https://www.atlassian.com/software/confluence

Atlassian. (n.d.b.). Confluence – Features.
Haettu 4.11.2017 osoitteesta
https://www.atlassian.com/software/confluence/features

Atlassian. (n.d.c.). Confluence Plugin Guide.
Haettu 9.11.2017 osoitteesta
https://developer.atlassian.com/confdev/confluence-plugin-guide

Atlassian. (n.d.d.). Writing Confluence Plugins.
Haettu 9.11.2017 osoitteesta
https://developer.atlassian.com/confdev/confluence-plugin-guide/writing-confluence-plugins

Atlassian. (n.d.e.). Atlassian Developers – Getting Started.
Haettu 10.11.2017 osoitteesta
https://developer.atlassian.com/docs/getting-started

Atlassian. (n.d.f.). Creating your Plugin Descriptor.
Haettu 19.11.2017 osoitteesta
https://developer.atlassian.com/confdev/confluence-plugin-guide/writing-confluence-plugins/creating-your-plugin-descriptor

Atlassian. (n.d.g.). Adding Plugin and Module Resources.
Haettu 19.11.2017 osoitteesta
https://developer.atlassian.com/confdev/confluence-plugin-guide/writing-confluence-plugins/adding-plugin-and-module-resources

Atlassian. (n.d.h.). Web Resource Module.
Haettu 20.11.2017 osoitteesta
https://developer.atlassian.com/confdev/confluence-plugin-guide/confluence-plugin-module-types/web-resource-module

Atlassian. (n.d.i.). Macro Module.
Haettu 26.12.2017. osoitteesta
https://developer.atlassian.com/server/confluence/macro-module/

Atlassian. (n.d.j.). Servlet Module.
Haettu 26.12.2017 osoitteesta
https://developer.atlassian.com/server/confluence/servlet-module/

Atlassian. (n.d.k.). Working with the SDK.
Haettu 1.1.2018 osoitteesta
https://developer.atlassian.com/server/framework/atlassian-sdk/working-with-the-sdk/

Bloomberg. (n.d.). Company Overview of Atlassian Pty Ltd.
Haettu 9.11.2017 osoitteesta
https://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=10655306

Kohler, S. (2013). Atlassian Confluence 5 Essentials. Packt Publishing.

Oracle. (n.d.a.). Lesson: Packaging Programs in JAR Files. Haettu 9.11.2017 osoitteesta
https://docs.oracle.com/javase/tutorial/deployment/jar/index.html

Oracle. (n.d.b.). Using JAR Files: The Bas.ics.
Haettu 9.11.2017 osoitteesta
https://docs.oracle.com/javase/tutorial/deployment/jar/bas.icsindex.html

Oracle. (n.d.c.). What is a Servlet?
Haettu 27.12.2017 osoitteesta
https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html

Sandoval, J. (2009). RESTful Java Web Services. Packt Publishing.

Sharpened Productions. (n.d.). SDK Definition.
Haettu 10.11.2017 osoitteesta
https://techterms.com/definition/sdk

Synacor. (n.d.a.). Open Source Email Platform - Zimbra Collaboration Open Source Edition. Haettu 1.11.2017 osoitteesta
https://www.zimbra.com/open-source-email-overview/

Synacor. (n.d.b.). Zimbra REST API Reference.
Haettu 1.11.2017 osoitteesta
https://wiki.zimbra.com/wiki/Zimbra_REST_API_Reference

Synacor. (n.d.c.). Zimbra REST API Method: Get Calendar.
Haettu 1.11.2017 osoitteesta
https://wiki.zimbra.com/wiki/Zimbra_REST_API_Reference:Get_Calendar

Synacor. (n.d.d.). Zimbra REST API Method: Get FreeBusy.
Haettu 1.11.2017 osoitteesta
https://wiki.zimbra.com/wiki/Zimbra_REST_API_Reference:Get_FreeBusy

Synacor. (n.d.e.). Zimbra REST API Method: Import Appointments.
Haettu 1.11.2017 osoitteesta
https://wiki.zimbra.com/wiki/Zim-
bra_REST_API_Reference:Import_Appointments